MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

DEFENSE COMMUNICATIONS ENGINEERING CENTER

TECHNICAL NOTE NO. 21-81

ANALYSIS OF PULSE STUFFING MULTIPLEXERS VIA DIGITAL COMPUTER SIMULATION WITH APPLICATION TO THE AN/GSC-24 ASYNCHRONOUS TDM

MAY 1981

DTIC

WA128895

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
DCEC TN 21-81	A128 895	

4. TITLE (and Subtitle)	5. TYPE OF REPORT & PERIOD COVERED
Analysis of Pulse Stuffing Multiplexers via Digital Computer Simulation With Application to the AN/GSC-24 Asynchronous TDM	Technical Note
	6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)	8. CONTRACT OR GRANT NUMBER(s)
David R. Smith John J. Cormack	

9. PERFORMING ORGANIZATION NAME AND ADDRESS	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
Defense Communications Engineering Center, R220 1860 Wiehle Avenue, Reston, VA 22090	N/A

11. CONTROLLING OFFICE NAME AND ADDRESS	12. REPORT DATE
Defense Communications Engineering Center, R220 1860 Wiehle Avenue, Reston, VA 22090	May 1981
	13. NUMBER OF PAGES
	101

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report)
N/A	Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Review for relevance 5 years from submission

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Pulse Stuffing	Slew Rate
Asynchronous	Jitter
AN/GSC-24 Asynchronous TDM	Synchronization
Time Division Multiplexing	Computer Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Results of computer simulation of asynchronous pulse stuffing multiplexers are presented. Analysis and simulation are given for output jitter, characterized by a continuous change in the output frequency as the demultiplexer attempts to smooth the effects of deleting stuff (dummy) bits from the received signal. The effect of jitter is shown on terminal equipment where narrowband bit synchronizers cannot track the jitter and will experience degraded performance including periodic loss of bit synchronization. Emphasis is placed on (over)

the AN/GSC-24 ATDM because of numerous interface modems which have arisen due to pulse stuffing jitter. Results of GSC-24 simulation are compared with actual test results. Simulation is shown to be effective in predicting performance for a variety of port and multiplex configurations.

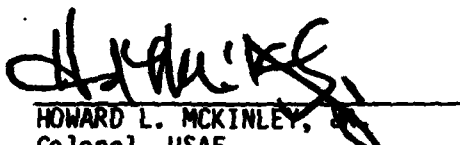| Accession For	
NTIS GRA&I	X
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

TECHNICAL NOTE NO. 21-81

ANALYSIS OF PULSE STUFFING MULTIPLEXERS VIA
DIGITAL COMPUTER SIMULATION WITH APPLICATION
TO THE AN/GSC-24 ASYNCHRONOUS TDM

MAY 1981

Prepared By:

● John J. Cormack

● David R. Smith

Approved for Publication:

HOWARD L. MCKINLEY,
Colonel, USAF
Chief, Transmission
Engineering Division

FOREWORD

EXECUTIVE SUMMARY

This report provides analysis via computer simulation of pulse stuffing multiplexers and their interface with various terminal equipment. The most important interface characteristic of a pulse stuffing multiplexer is its output jitter, characterized by a continuous change in the output frequency as the demultiplexer attempts to smooth the effects of deleting pulse (dummy) bits from the received signal. The effect of this frequency variation is that certain terminal equipment with narrowband clock recovery loops cannot track the jitter and will experience degraded performance including periodic loss of bit synchronization.

The focus of the analysis is on the AN/GSC-24 asynchronous time division multiplexer (ATDM), one of first pulse stuffing multiplexers used within the DCS. Because of a flexible input/output capability, the AN/GSC-24 has found widespread application in the DSCS, thus requiring interface with a variety of low, medium and high speed terminal and higher level multiplex equipment. Because of numerous interface problems which have arisen in recent months with the AN/GSC-24, the need for interface testing has been established and several tests have been conducted at MilDep test facilities. The need for analysis and computer simulation was recognized as a means of supplementing current and future testing and providing insight into the performance of the AN/GSC-24.

The simulation, written in the GASP simulation language, allows modeling of a pulse stuffing multiplexer, including structuring of the TDM frame, selection of values for pulse stuffing parameters, and selection of a bit synchronization type as would be found in an interfacing terminal equipment. Specifically, for the AN/GSC-24, one can insert stuff and/or fill bits into the multiplexer at various rates and remove these stuff/fill bits from the demultiplexer output. The simulation output consists of time domain waveforms of the demultiplexer's smoothing loop response or terminating equipment's bit synchronizer response, from which jitter performance data can be determined. For those parameters of interest to typical AN/GSC-24 applications, simulation runs have been made and results tabulated or plotted. Performance data are given for low speed applications of the AN/GSC-24 as a function of user rate offsets, multiplex configuration (with and without fill bits), smoothing buffer strap settings, bit synchronizer type, and for tandem connections. Comparisons of these results with analytical predictions and actual test results are also given.

Such simulation results can greatly reduce the test and evaluation time for AN/GSC-24 equipment interfaces and for other pulse stuffing multiplexers (e.g., DRAMA, LSTDM, and Adaptive Multiplexer). Additionally, insight is provided into the reasons for the observed AN/GSC-24 performance. One significant result of this new insight will be the improvement of AN/GSC-24 multiplex configurations, since jitter performance can now be characterized for any choice of channel/port rates and an optimum choice of multiplexer configurations can be made accordingly.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Cont'd)

LIST OF TABLES

I. INTRODUCTION

Initial implementations of digital transmission in the DCS, such as the Frankfurt-Koenigstuhl-Vaihingen (FKV) Project and the Digital European Backbone (DEB) Stage I, are using pulse stuffing as the basic synchronization technique. The existing AUTOSEVOCOM network, which is largely based on 50 kb/s KY-3 operation, is accommodated via a pulse stuffing interface with first level multiplexers. The Digital Communications Subsystem (DCSS) of the DSCS employs a pulse stuffing multiplexer, the AN/GSC-24 asynchronous TDM, for all levels of digital multiplexing. Provision of this pulse stuffing interface makes it possible to use the medium grade clocks typically found in terminal and transmission equipment and thereby eliminates the need for a separate, highly accurate frequency source. Since major portions of the terrestrial and satellite DCS subsystems will continue to utilize pulse stuffing for synchronization in the foreseeable future, it is important to understand the properties of pulse stuffing interfaces and anticipate problems with a pulse stuffing network.

The pulse stuffing process as used in digital multiplexing inserts stuff bits periodically to adjust all incoming channels to a common bit rate. The location of each stuff bit is communicated to the far-end companion demultiplexer. Here the stuff bits are identified and deleted to return each channel bit stream to its original rate. In the process, pulse stuffing jitter is introduced, which results from the inability of the demultiplexer to provide perfectly smoothed clock and data outputs. This phenomenon can cause timing problems in downstream equipments which must extract timing from the jittered clock and data signals. A narrowband filter, as found with most bit synchronizers, may not be able to track the high frequency jitter inherent in most destuffed clock signals. Such problems led to a DCA R&D contract for a study of pulse stuffing interfaces [1]. From that study came recommendations for specifying a standard interface between pulse stuffing multiplexers and bit synchronizers.

This technical note analyzes pulse stuffing multiplexers with the objective of giving insight into the phenomenon of slewing jitter, predicting performance of various interfaces, and guiding future test efforts. The tools utilized are theoretical analysis where possible but computer simulation for most applications. A digital computer simulation of pulse stuffing multiplexers was previously developed under the aforementioned R&D contract [1]. This program has been expanded by the authors to include the capability of modeling the AN/GSC-24 ATDM.

Because of numerous interface problems experienced with AN/GSC-24 operation, the analysis herein has been focused on the AN/GSC-24. The main problem observed in various interface tests has been excessive jitter in AN/GSC-24 demultiplexer outputs and resultant degraded performance with interfacing equipment. The simulation used herein to study this interface problem allows the structuring of the AN/GSC-24 frame including stuff and fill bits, selection of values for the demultiplexer smoothing loop, and selection of bit synchronization parameter values for the interfacing equipment.

1

To run the AN/GSC-24 simulation, the stuff/fill bit locations are determined from a multiplex plan, which together with smoothing loop parameters are input to the simulation. The computer program, which uses the GASP simulation language, generates time domain plots of the smoothing loop output. Additionally, the program allows interface of the smoothing loop output with a bit synchronizer and can provide time domain plots of the bit synchronizer response. This latter feature thus simulates a test of the AN/GSC-24 with any given terminal equipment. Finally, performance in a tandemed multiplexer connection is simulated, thus accounting for networking applications of the AN/GSC-24.

The following sections first describe theoretically the design, operation, and performance of pulse stuffing multiplexers. A more thorough description of the AN/GSC-24 is then given, including multiplex, demultiplex, and smoothing loop functions. The simulation program is generally described, which, along with several appendices, allows the interested reader to further use the program in studying the subject problem. Finally, results of AN/GSC-24 simulation runs are presented for a number of configurations of interest to DCS/DSCS applications.

The choice of simulating and studying the AN/GSC-24 multiplexer stemmed from interface problems which arose when the AN/GSC-24 was tested in the laboratory, or in some cases, when actually implemented in the field. The advantage of the simulation described herein is that interfaces with pulse stuffing multiplexers planned for use in the DCS can be verified before final system design. Any interface problems uncovered by the simulation would allow changes to equipment or system design before final test and implementation. The savings in time and money are potentially great. There are a number of pulse stuffing interfaces that are planned for the DCS, in addition to the existing systems mentioned earlier. DCS networks such as DEB, Data Transmission Network (DTN), and the ECCM Network all will utilize pulse stuffing interfaces. The pulse stuffing simulation can be used to analyze these networks for anticipated interfaces, for both individual equipments and for end-to-end connections. Indeed, such an analysis is planned for the pulse stuffing design found in the DRAMA AN/FCC-99 multiplexer.

II. THEORY OF PULSE STUFFING MULTIPLEXERS

1. DESCRIPTION

A time division multiplexer (TDM) may be categorized as providing either synchronous or non-synchronous (or asynchronous*) interface with each input data channel. A synchronous interface specifies that each source which interfaces the TDM must be timed by the same clock which provides timing within the TDM. For this case, there is no ambiguity between the arrival time of each data channel bit and the time to multiplex each data channel bit. However, if each data source is timed by its own internal clock, the interface is said to be non-synchronous. For this case, slight differences between the data source clock and TDM clock will arise due to the clock inaccuracies and instabilities; the two clocks will then wander apart and eventually will cause a timing bit to be added or deleted in the TDM, resulting in loss of synchronization. To compensate for this lack of synchronization between each source and the TDM, some technique is required to convert each non-synchronous input to a rate which is synchronous with the TDM clock frequency. The general approach to asynchronous/synchronous conversion is illustrated in Figure 1. Each data source is timed at some nominal rate f_i which will range to $f_i \pm \Delta f_i$, where the size of the window $\pm \Delta f_i$ is dependent on clock performance parameters but is always much smaller than f_i. The TDM must be capable of accepting each nominal input rate over its full range and must provide a conversion of each input data channel to a rate which is synchronous with the internal clock of the TDM. Figure 1 shows a conversion from the nominal data channel frequency f_i to a frequency f_i' for synchronous multiplexing in the TDM. Two commonly applied techniques for accomplishing this asynchronous to synchronous conversion are described below.

2. POSITIVE PULSE STUFFING.

In the positive pulse (or bit) stuffing technique, each input data channel rate f_i is synchronized to a TDM channel rate which is at a higher rate than the maximum deviation of the input data channel ($f_i + \Delta f_i$). Figures 2 and 3 show typical block diagrams of the transmit and receive sections, respectively, of a positive pulse stuffing TDM. Input channel data is written into an elastic buffer at a rate of $f_i + \Delta f_i$, and read out of the buffer at a rate f_i'. Since f_i' is at a rate slightly higher than $f_i + \Delta f_i$, there is a tendency to deplete the buffer contents. Hence the buffer fill is monitored and compared to a preset threshold. When this threshold is reached, a stuff (dummy) bit is requested. At the next available stuff opportunity, the read clock is inhibited for a single clock pulse allowing a stuff bit to be inserted into the outgoing data. Simultaneously, the precise location of the stuffed time slot is coded into the overhead

---

* The term "asynchronous" here will also be used to mean non-synchronous.
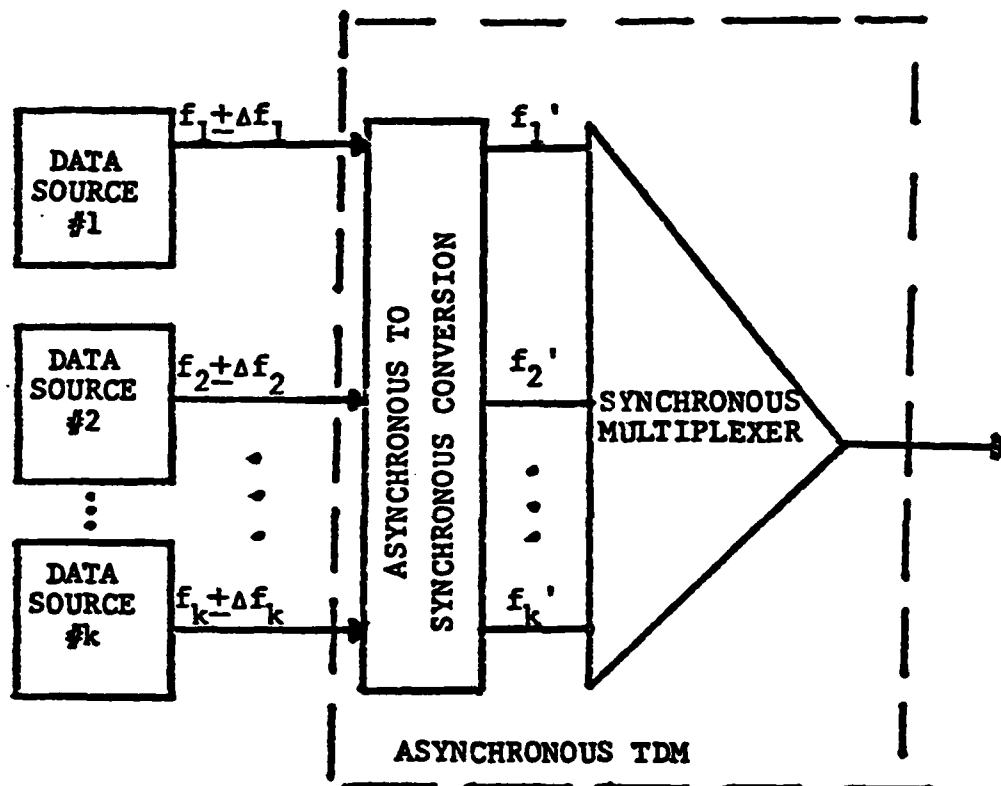
Figure 1.   Asynchronous TDM
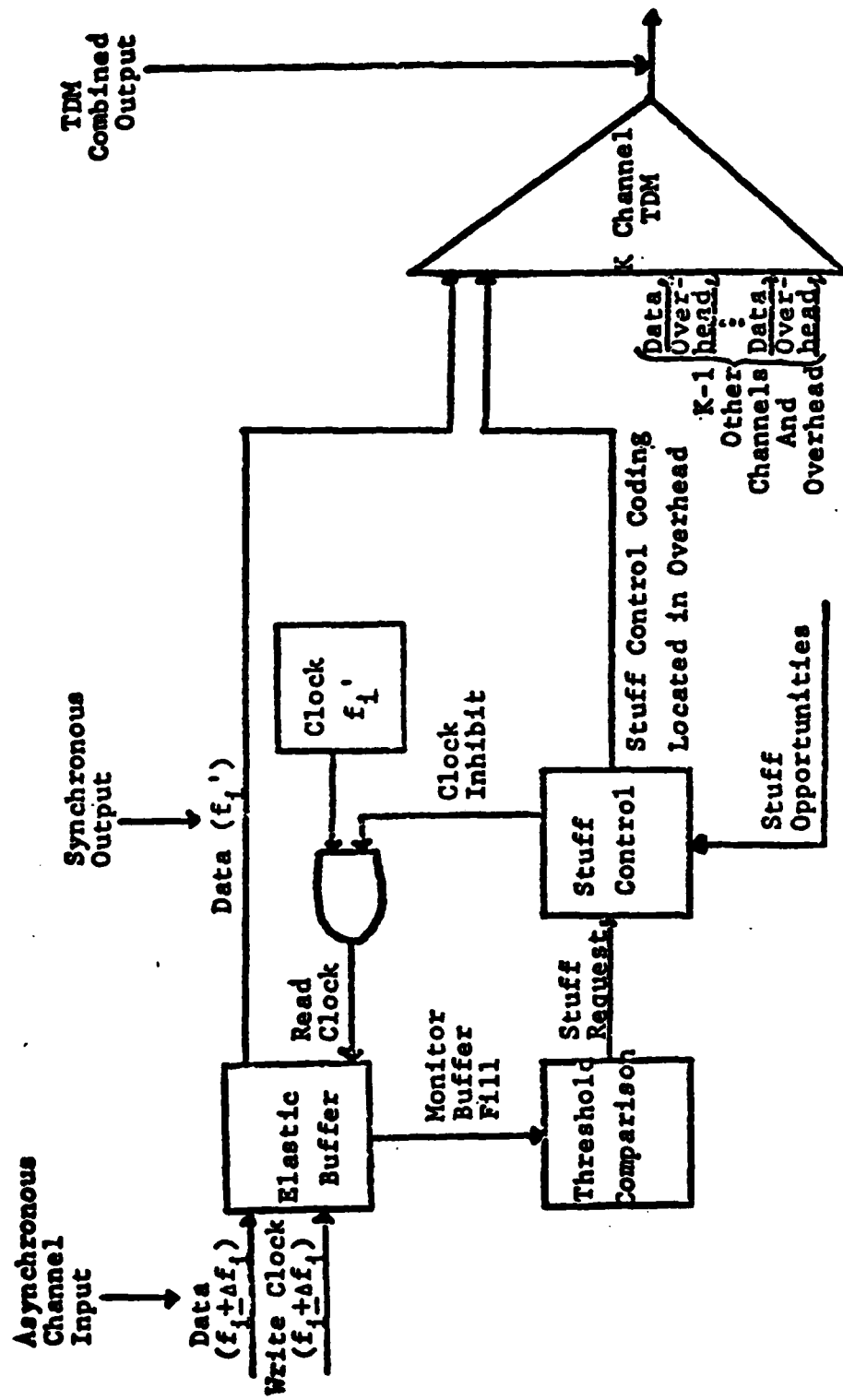
Figure 2. Positive Bit Stuffing Synchronization - Transmit Section
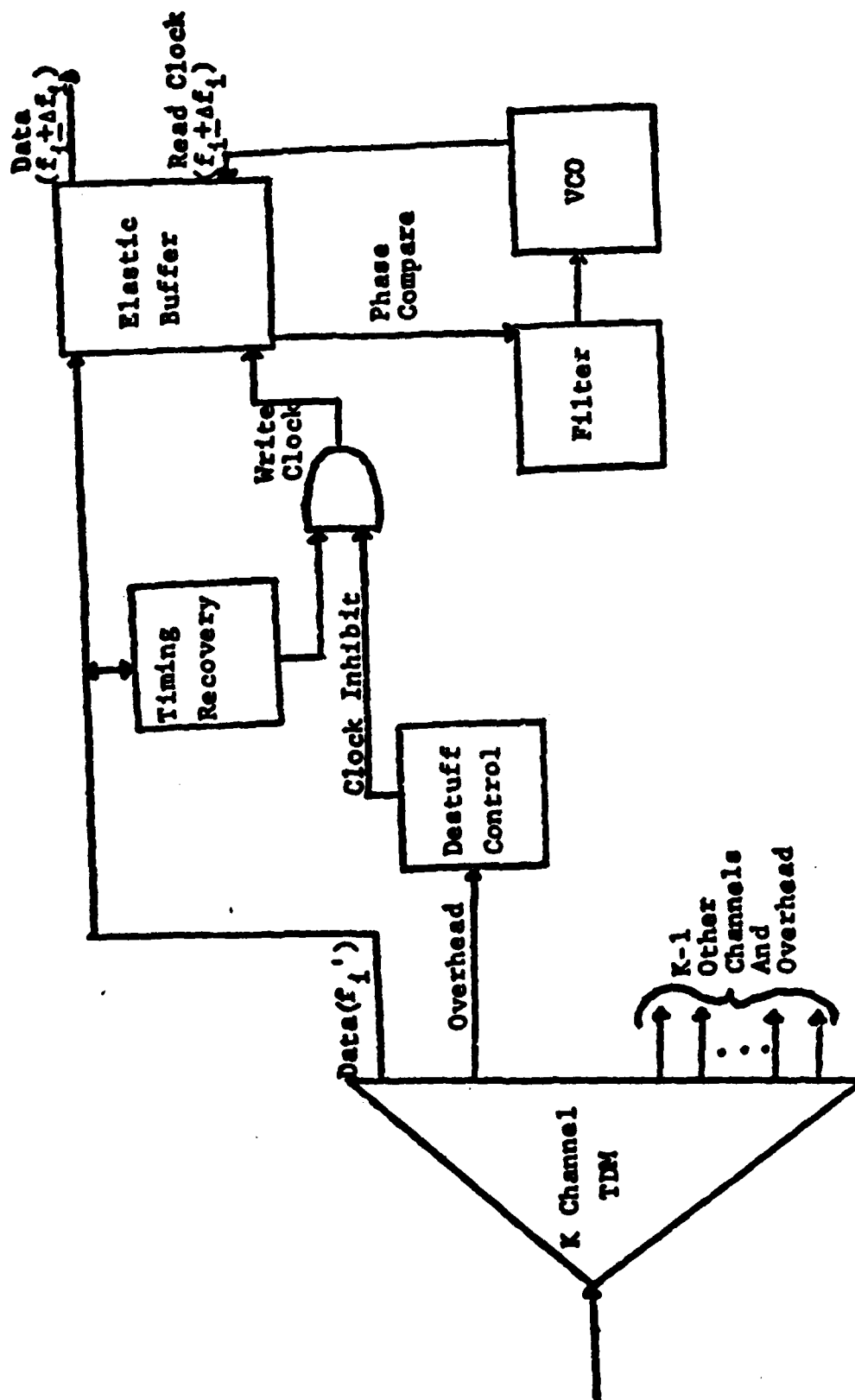
Figure 3. Positive Bit Stuffing Synchronization. Receive Section

6

channel and transmitted to the receive end (demultiplexer) of the TDM. At the receive end, shown in Figure 3, the received channel data is written into an elastic buffer, but the stuff bits are inhibited from the overhead channel which drives a clock inhibit circuit. Data is read out of the buffer by a smooth clock. Clock smoothing is achieved by monitoring buffer fill to drive a smoothing phase-locked loop (filter plus voltage controlled oscillator).

3. POSITIVE-NEGATIVE PULSE STUFFING

In a positive-negative pulse stuffing technique, each input data channel rate $f_i + \Delta f_i$ is synchronized to a TDM channel rate f_i which is identical to the nominal input data channel rate. Buffering of the input is provided just as shown in Figure 2. However, now the buffer contents may deplete or spill, since the buffer is read out at a frequency f_i, whereas the input data rate may have either negative or positive offset about the nominal frequency f_i. If the data source rate is low, the buffer will tend to deplete and for this case stuff bits are added (positive stuff); if the input rate is high, the buffer will tend to spill and here channel bits are subtracted (negative stuff) from the input rate to match the TDM rate. For the positive-negative bit stuffing, two actions may occur with each stuff opportunity: (1) positive stuff or (2) negative stuff. In a positive-zero-negative bit stuffing scheme, three actions may occur with each stuff opportunity: (1) no stuff, (2) positive stuff, or (3) negative stuff. Stuff locations are coded into an overhead channel to allow a receiving end to properly remove positive stuff bits and add negative stuff bits.

4. BIT COUNT INTEGRITY PERFORMANCE

The receive end of the TDM must properly locate each stuff bit in order to delete positive stuff bits, add negative stuff bits, and derive each original asynchronous source. Incorrect decoding of a stuff bit due to transmission errors will cause loss of bit count integrity in the channel associated with the incorrectly decoded command, since the demultiplexer will have incorrectly added or deleted a bit in the derived asynchronous channel. In order to protect against potential loss of synchronization, each stuff location is redundantly signaled with a code, where two actions (stuff/no stuff) are signalled for a positive bit stuffing TDM and where three actions (no stuff/positive stuff/negative stuff) are signalled for a positive-negative bit stuffing TDM. After transmission, majority logic applied to each code word determines the proper action. The performance measure is then given by the probability of incorrect decoding per stuff action, and, from this probability, the time to loss of synchronization due to incorrect decoding of a stuff may be obtained. If we assume independent probability of error from code bit to code bit, then the probability of incorrect decoding for a particular stuff action is given by the binominal probability distribution.

7

$$P(SWE) = \text{Prob[stuff word error]} = \sum_{i=\frac{N+1}{2}}^{N} \binom{N}{i} p^i (1-p)^{N-1} \tag{1}$$

where

N = code length

p = probability of bit error over transmission channel.

For small probability of bit error, this expression can be approximated by

$$P[SWE] = \binom{N}{x} p^x (1-p)^{N-x} \tag{2}$$

where $x = (N+1)/2$. Given the statistics (probability density function) of the probability error, p, one can find the mean $\bar{p}$, and then use $\bar{p}$ to find a mean P(SWE).[2] Likewise, the variance of p can be found from statistics of p, and a variance of P(SWE) can then be stated. Knowing the rate of stuffing opportunities (R, given in stuff/unit time), the mean and variance of the time to loss of BCI due to stuff word error can be calculated as the

$$\text{Mean Time to Loss of BCI} = \frac{1}{R \cdot \overline{P(SWE)}} \tag{3}$$

$$\text{Variance of Time to Loss of BCI} = \frac{1}{R \cdot \sigma^2_{P(SWE)}}. \tag{4}$$

Application of these equations to specific examples is shown in reference [2].

5. JITTER PERFORMANCE.

The removal of the stuffed bits at the demultiplexer causes timing jitter at the individual channel output, which has been called pulse stuffing jitter. The actual stuffing rate occurs at average frequency of f_0 stuffs per second and is equal to the difference between the multiplexer synchronous rate and the channel input rate. In addition, since there is a delay or waiting time between the initiation of a stuff request and the time at which the stuff is actually accomplished, the spacing between stuffed bits varies about the average value of $1/f_0$ seconds, resulting in waiting time jitter at the demultiplexer output. The phase-locked loop generating the read clock acts to reduce this jitter. The system designer must choose f_0 large enough and the loop bandwidth narrow enough so that the jitter is reduced to an acceptable value (i.e., at a low enough frequency so that a bit synchronizer in downstream equipment is not affected). There is also jitter introduced by the insertion and removal of overhead bits (stuff bits, control information, frame bits) that must be attenuated by the phase-locked loop.

The asynchronous input of a positive stuffing multiplexer consists of a data stream whose frequency is less than the synchronous output data stream. The difference between these two rates is justified by discrete one bit jumps at the synchronous output. The timing relationship between the asynchronous input and synchronous output is shown in Figure 4. In this figure each output bit has a period which is 4/5 the period of the input bit so that the instantaneous output rate is 1.25 times the input rate. Every fifth bit of the synchronous output is converted to a stuff (dummy) bit. Because of the rate difference, the phase of the output stream increases linearly with respect to the input stream. When this phase difference accumulates to one bit, the phase is reset by inserting a dummy bit in place of an input data bit. Pulse stuffing jitter is therefore a sawtooth waveform with a peak-to-peak value of one bit interval and a period equal to the stuffing rate. The slope of the sawtooth equals the frequency offset between the input asynchronous rate and the synchronous output rate.

a. Waiting Time Jitter. For the jitter waveform to be the periodic sawtooth described above, the stuff bit needs to be inserted at the specific instant that the phase error crosses a threshold. However, only discrete time slots are available for stuff bits, at the stuff opportunity rate, f_s, and if phase threshold crossings do not occur at these times an additional jitter component is generated. This so-called waiting time jitter [3] also forms a sawtooth waveform as indicated in Figure 5. The resulting frequency will generally be so low that the smoothing filter will have little effect on reducing the peak of this component of jitter. The peak waiting time jitter may be found from Figure 6. If the phase error crosses the threshold immediately after an available stuff opportunity, then the error will accumulate until the next possible stuff opportunity. The maximum amount of phase error which can accumulate during this interval equals the frequency offset times the interval between stuff opportunities,

$$\phi_\epsilon = f_0 \, t_S \text{ bits} \tag{5}$$

where $t_S = 1/f_S$ sec.

The stuffing ratio, ρ, is defined as the ratio of the actual stuffing rate to the stuffing opportunity rate, or

$$\rho = \frac{f_0}{f_S} \leq 1. \tag{6}$$

The unfiltered peak-to-peak waiting time jitter can then be expressed as

$$\phi_\epsilon = f_0 t_S = \rho f_S t_S = \rho \text{ bits} \tag{7}$$

9
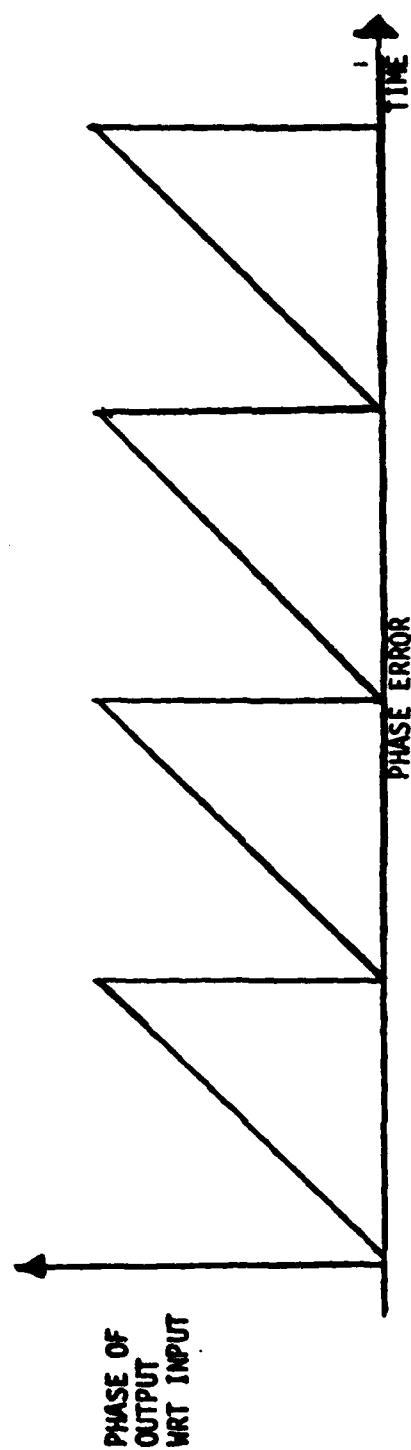
Figure 4. Asynchronous/Synchronous Timing Relationships

AVAILABLE PULSE STUFFING POSITIONS

PHASE ERROR WHEN THRESHOLD CROSSINGS CORRESPOND
TO AN AVAILABLE PULSE STUFFING SLOT
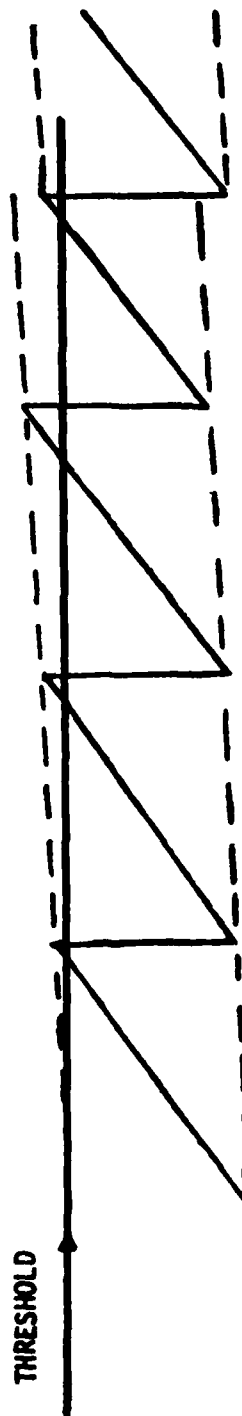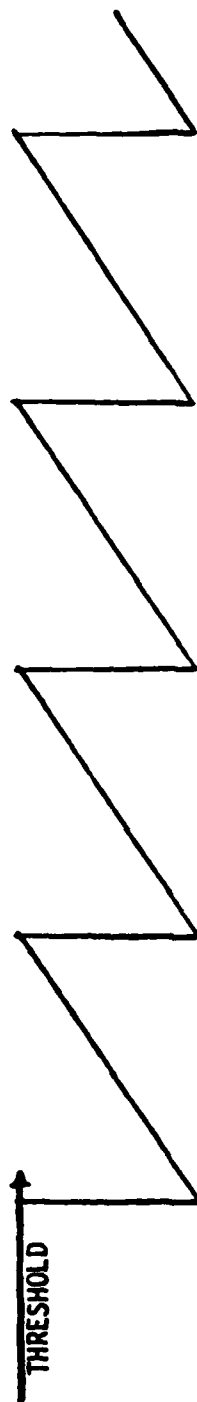
THRESHOLD

PHASE ERROR WHEN THRESHOLD CROSSINGS DO NOT
CORRESPOND TO AVAILABLE PULSE STUFFING SLOTS;
THE DASHED ENVELOPE INDICATES THE LOW
FREQUENCY COMPONENT OF THE PHASE ERROR

THRESHOLD

Figure 5. Waiting Time Jitter

11
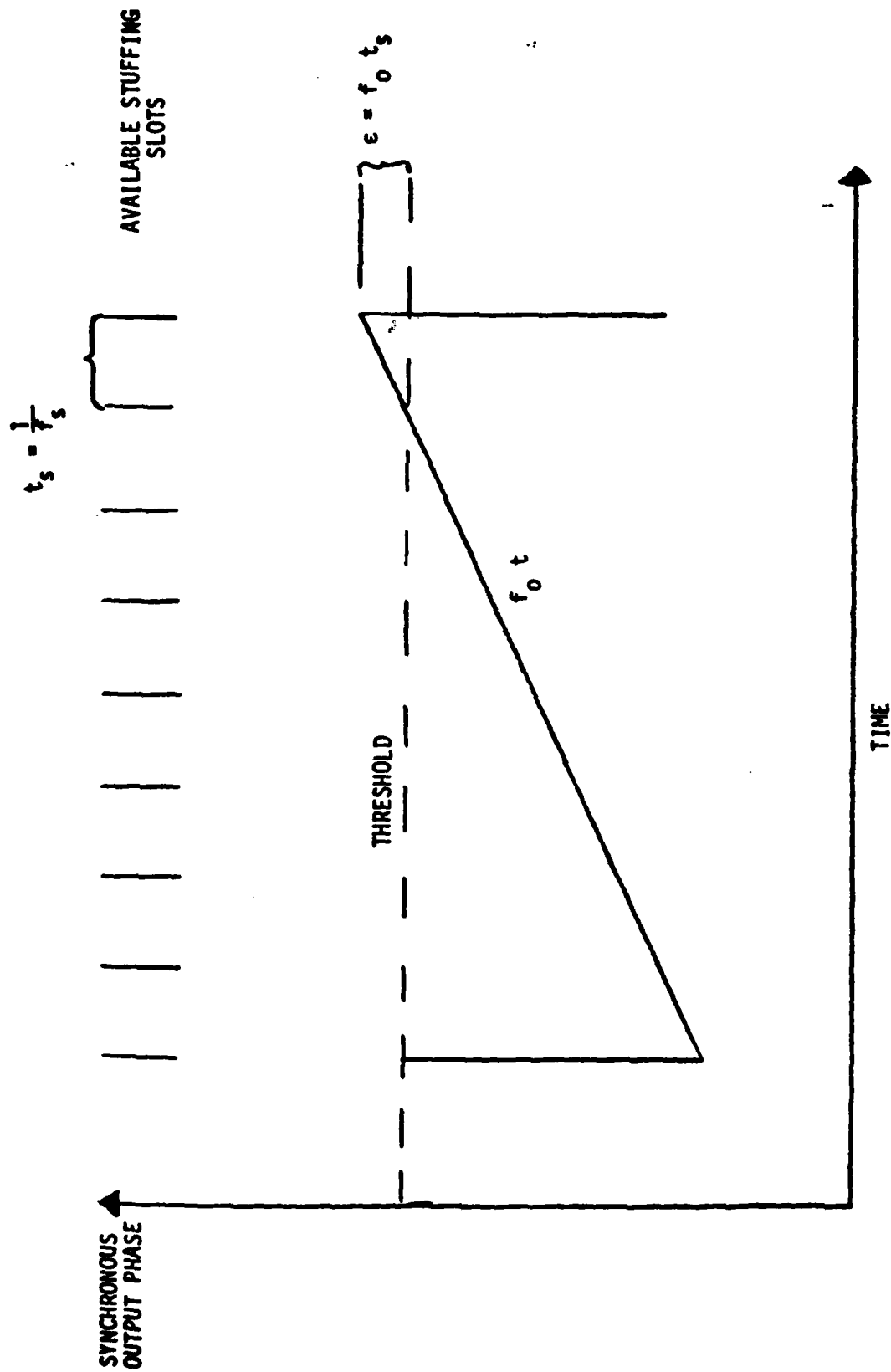
Figure 6. Peak Value of Waiting Time Jitter

Figure 7 plots the peak-to-peak amplitude of the filtered waiting time jitter [3]. Although the jitter should be directly proportioned to ρ, the smoothing filter has some effect on jitter amplitude, particularly at lower stuffing ratios where the smoothing filter does not significantly attenuate the jitter. At $\rho = 1$, peak-to-peak waiting time jitter is 1 bit and its frequency is so low that the smoothing loop doesn't attenuate it. At $\rho = 1/2$, the peak-to-peak jitter equals 1/2 bit (6 dB below value at $\rho = 1$); likewise, at $\rho = 1/3$, 1/4, and 1/5, the peak-to-peak jitter equals 1/3 (9.6 dB), 1/4 (12 dB), and 1/5 (14 dB) bits. In general, for $\rho = 1/n$, the peak-to-peak jitter equals 1/n. The conclusion reached from Figure 7 is that the waiting time jitter peaks for rational values of n (with $\rho = 1/n$). Hence waiting time jitter can be minimized by selecting a non-rational n for the stuffing ratio. However, the alternative of lowering the stuffing ratio to minimize jitter would cost additional overhead and further jeopardize bit count integrity performance.
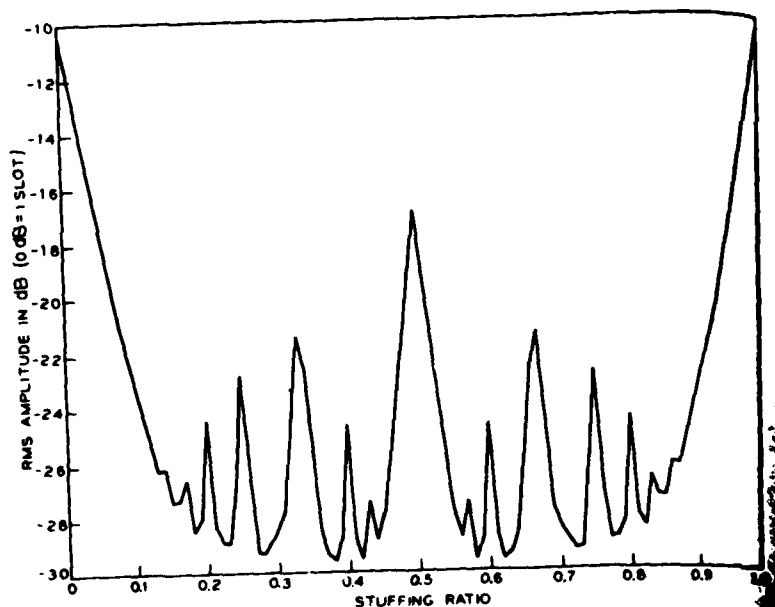
Figure 7. Waiting Time Jitter vs Stuffing Ratio

13

b. <u>Effect of Smoothing Filter on Pulse Stuffing Jitter.</u> Although waiting time jitter will generally occur at a low frequency so that it cannot be removed by smoothing, the stuffing jitter component can be attenuated by narrowband smoothing loops, as shown in Figure 8.
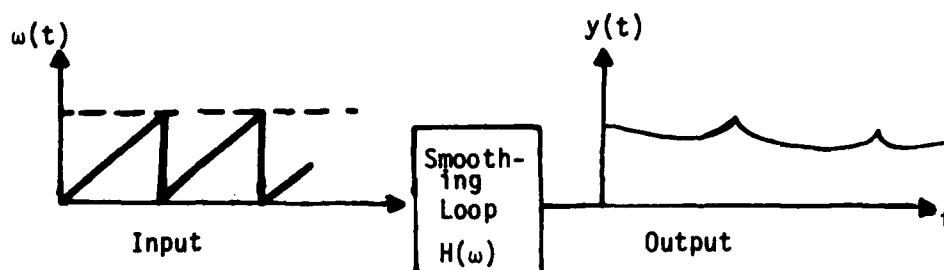
Figure 8. Model of Smoothing Loop Effect on Jitter

The smoothing loop output waveform may be computed by convolving the loop impulse response with the sawtooth phase jitter waveform which represents the unsmoothed jitter. For a first order loop the closed loop transfer function is

$$H(\omega) = \frac{\omega_c}{\omega_c + j\omega} . \tag{8}$$

The corresponding impulse response is

$$h(t) = 2\pi f_c e^{-2\pi f_c t} , \tag{9}$$

where f_c is the 3 dB cutoff frequency of the smoothing filter. The periodic sawtooth waveform can be written as

$$\omega(t) = t - n \qquad -0.5 + n < t < 0.5 + n \qquad n = 0, \pm 1, \pm 2, \dots \tag{10}$$

14

where the period has been normalized by the reciprocal of the stuffing rate, f_0, and the sawtooth peak-to-peak amplitude has been normalized by 360° (1 bit). The result of the convolution of the first order smoothing loop with the periodic sawtooth is

$$y(t) = \int_{\infty}^{t} \omega(\alpha) \, 2\pi f_c e^{k-2\pi f_c(t-\alpha)} \, d\alpha \, . \tag{11}$$

Upon carrying out the integration, the output jitter over the period $-0.5 < t < 0.5$ is

$$y(t) = t - \frac{1}{2\pi f_c} + e^{-2\pi f_c t} \left(\frac{e^{-\pi f_c}}{1-e^{-2\pi f_c}} \right). \tag{12}$$

The phase jitter $y(t)$ given by Equation (12) is plotted in Figure 9. Note that the smoothing loop with narrower bandwidth provides greater jitter attenuation. Another important pulse stuffing multiplexer characteristic is the slew rate, i.e., rate of change of phase, which is of course the definition of frequency, given by

$$f(t) = \dot{y}(t). \tag{13}$$

Carrying out the differentiation,

$$f(t) = 1-2\pi f_c e^{-2\pi f_c t} \left(\frac{e^{-\pi f_c}}{1-e^{-2\pi f_c}} \right) \tag{14}$$

A plot of $f(t)$ is given in Figure 10 for selected values of smoothing loop bandwidth. Again note that greater slew rate attenutation is provided by narrower smoothing loop bandwidths.

     c. Effect of Jitter on Bit Synchronizers. A significant problem in integrating a pulse stuffing multiplexer into a system which contains bit synchronizers is the effect that jitter and slew rate have on timing recovery circuits. Excessive jitter can cause a phase-locked loop (PLL) to lose lock or slip bits. Since timing derived by the PLL is used to make bit decisions, it is not enough that the loop merely maintain lock. It must also track instantaneous bit timing or bit decisions will be made with a timing error. For example, with matched filter demodulation of NRZ data, a timing error of one-quarter bit corresponds to a 6 dB loss in signal-to-noise-ratio, and a timing error of 0.1 bit corresponds to a 1.9 dB loss in SNR. The ability of the bit synchronizer to track input jitter is improved by increasing the loop bandwidth but at the expense of added noise bandwidth, so that bit sync bandwidths are generally no greater than necessary for proper timing recovery.

TIME NORMALIZED TO PERIOD OF JITTER
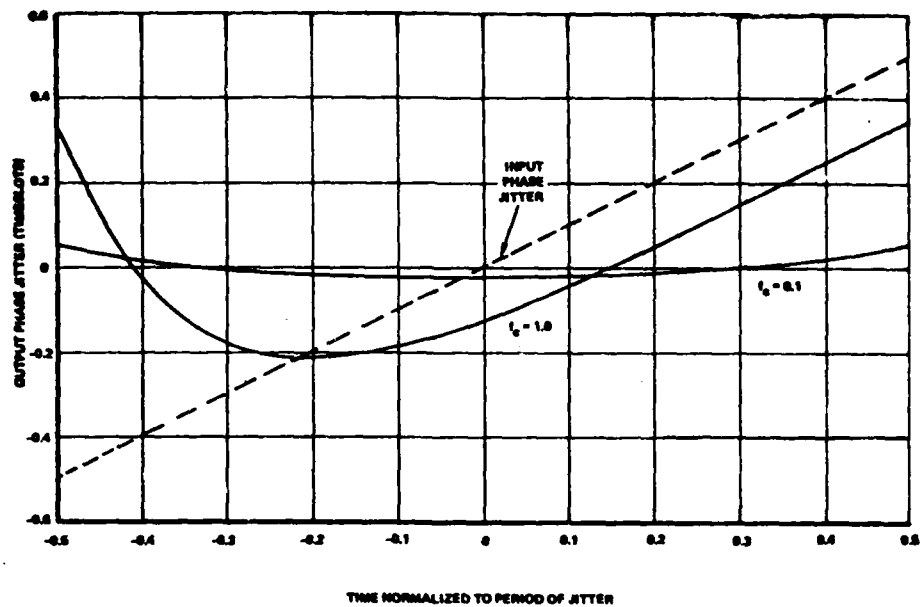
Figure 9.   Stuffing Jitter Waveform at Output
of First Order Smoothing Filter

TIME NORMALIZED TO PERIOD OF JITTER)
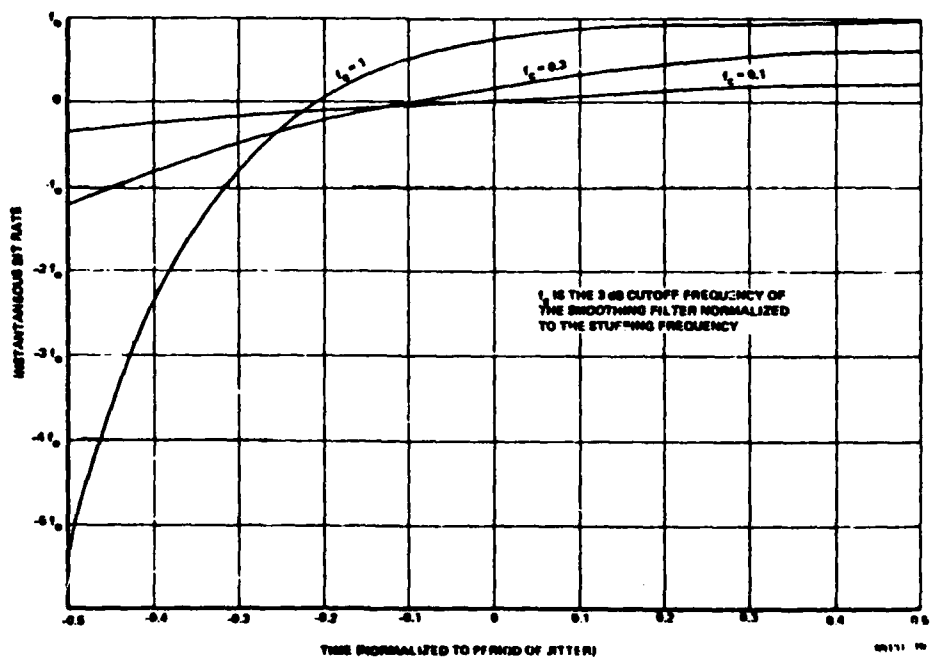
Figure   10.   Instantaneous Bit Rate Waveform at the
Output of a First Order Smoothing Filter
Attributed to Stuffing Jitter

16

In order to analyze the effect of pulse stuffing jitter on bit synchronizers, a model of a first order smoothing loop cascaded with a first order bit synchronizer has been developed, as shown in Figure 11. The transfer function of this cascade from input signal to bit sync phase detecter output is

$$H(\omega) = \frac{2\pi f_c}{2\pi f_c + (j\omega)} \cdot \frac{(j\omega)}{2\pi k f_c + (j\omega)} = \frac{2\pi f_c (j\omega)}{[2\pi f_c + j\omega][2\pi k f_c + (j\omega)]} \cdot \quad (15)$$

By partial fractions this expression can be expanded to

$$H(\omega) = \frac{1}{(1-K)} \left[\frac{2\pi f_c}{2\pi f_c + j\omega} - \frac{2\pi k f_c}{2\pi k f_c + j\omega} \right] \cdot \quad (16)$$

The expression is the linear sum of the first order loop expressions. We can convolve the sawtooth waveform representing the jitter input with this $H(\omega)$ as done before to arrive at

$$\theta_e(t) = \frac{1}{2\pi k f_c} + \frac{1}{1-k} \left(e^{-2\pi f_c t} \cdot \frac{e^{-\pi f_c}}{1-e^{-2\pi f_c}} - e^{-2\pi k f_c t} \cdot \frac{e^{-\pi k f_c}}{1-e^{-2\pi k f_c}} \right) \cdot (17)$$

This is the expression for phase error over one stuffing jitter period, $-0.5 < t < 0.5$. This waveform has a maximum value at the extremes ($t = \pm 0.5$) and a minimum point in-between. The minimum point is found by differentiating $\theta_e(t)$, setting equal to zero, and solving for t. The minimum point is located at

$$t = -1/2 - \frac{1}{2\pi f_c(1-k)} \ln \left(k \cdot \frac{1-e^{-2\pi f_c}}{1-e^{-2\pi k f_c}} \right) \cdot \quad (18)$$

The peak-to-peak phase error is then

$$\Delta\theta_{e_{p-p}} = \frac{1}{1-k} \cdot \frac{1}{1-e^{-2\pi f_c}} \left[1 - \frac{1-e^{-2\pi f_c}}{1-e^{-2\pi k f_c}} + \left(\frac{1-k}{k} \right) \cdot \left(k \cdot \frac{1-e^{-2\pi f_c}}{1-e^{-2\pi f_c k}} \right)^{\frac{1}{1-k}} \right] \cdot \quad (19)$$

θ_i

$+$ $-$ α $\frac{1}{s}$ $+$ $-$ β $\frac{1}{s}$ θ_o

$\alpha = 2\pi f_c$

$\beta = 2\pi k f_c$

$k = \beta/\alpha$

Smoothing Loop   Bit Synchronizer
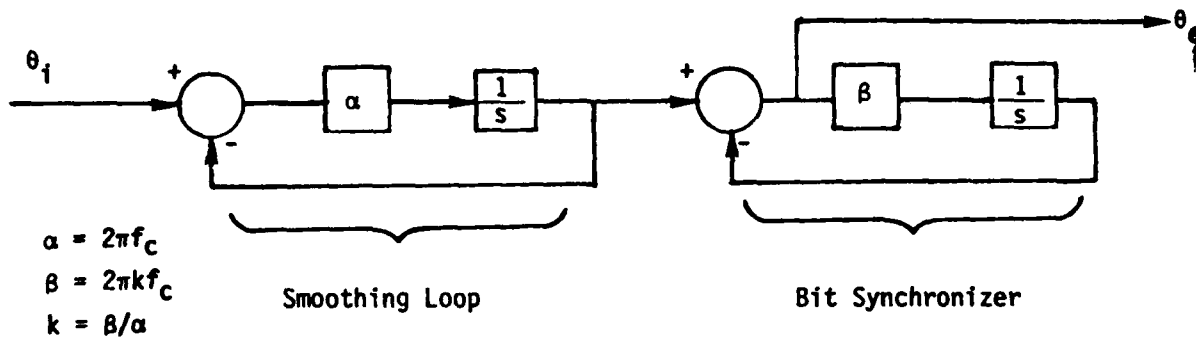
FIGURE 11.   Block Diagram of First Order Smoothing Loop
Cascaded with First Order Bit Synchronizer

Using Equation (19), curves are shown in Figure 12 for the peak-to-peak timing error when a first order smoothing loop is interfaced with a first order bit synchronizer. This figure presents the information for a constant ratio of bit synchronizer loop bandwidth to smoothing filter loop bandwidth. The curves only consider the effects of stuffing jitter since the analytical model did not consider waiting time jitter effects. To illustrate the significance of these curves, first note that when the smoothing filter loop has a bandwidth equal to the stuffing rate ($f_c = f_0$), the timing error is significantly decreased only when the bit synchronizer has a large bandwidth of approximately 10 times the stuffing rate. When the bit synchronizer bandwidth and the smoothing filter bandwidth are both equal to the stuffing rate ($k=1$, $f_c=f_0$), the timing error is approximately 30 percent of a bit, which corresponds to an 8.0 dB loss in signal-to-noise ratio.
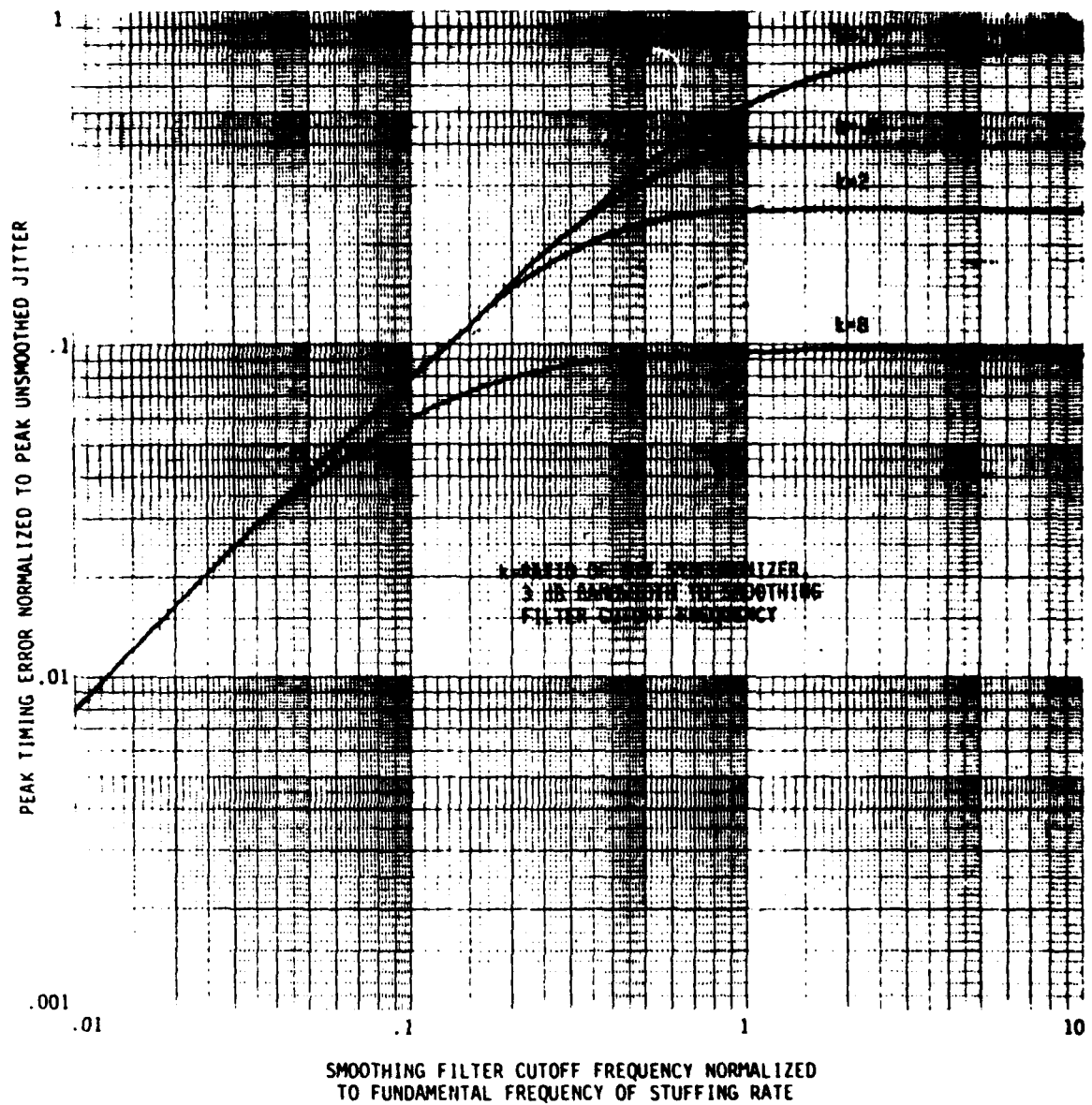
Figure 12. Effect of Pulse Stuffing Jitter on Data Recovery
Timing for a First Order Bit synchronizer

19

III. DESCRIPTION OF AN/GSC-24 ASYNCHRONOUS TDM

Here the operation of the AN/GSC-24 multiplexer, demultiplexer, and smoothing loop will be described. The AN/GSC-24 employs positive-zero-negative pulse stuffing. Each channel input has individual elastic buffers which have data written into them by the source clock and data read out from the buffer by the multiplexer output clock. Since the input and multiplexer rate elastic buffers are not synchronous, eventually the buffer would lose or gain a bit. At this time, a pseudo bit must be injected into or a data bit removed from the data stream to realign the phase relation of the data stream. This operation of inserting or removing bits must be controlled so the demultiplexer knows when to do the inverse operations. This knowledge is passed onto the demultiplexer via the overhead bits which convey the stuffing bit/data bit location.

1. MULTIPLEXER/DEMULTIPLEXER DESIGN

The GSC-24 design is structured around the use of ports for combining applied channel inputs into a single interleaved serial output data stream. A port is an interval of time during which data from a particular channel are allowed into the output stream. This rate is named the port rate, and once established for a given multiplexer configuration, remains constant for all ports in use. To enable the multiplexer to simultaneously accept input channels of different (mixed) rates, multiple ports are assigned to a single input channel. This arrangement is termed port strapping. For the GSC-24, the total number of assigned ports to a particular channel may vary between 1 and 25. The total number of assigned ports to all input channels of a given configuration is the sum of all individual channel values. One additional port is used within the multiplexer for the insertion of overhead data into the output bit stream. Thus the total number of ports, including overhead, used for any given configuration is $N + 1$. The multiplexer's output rate is the product of the total number of ports used ($N + 1$) and the port rate (R_p).

In general, a digital input channel rate will be nominally equal to the product of port rate and the number of ports assigned to that channel. For cases where this is not true, the multiplexer is capable of performing a rate conversion process termed COARSE RATE CONVERSION (CRC). The most important benefit resulting from the multiplexer's coarse rate conversion capability is that it enables simultaneous processing of digital channel input rates from mixed rate families into a single data stream.

When the channel sampling rate for a specific channel cannot be configured to be within a tolerance of ± 250 ppm of the incoming data rate, the channel sampling rate for that channel is set higher than the incoming data rate. The coarse rate conversion circuits then delete a predetermined number of gated clock signals to effectively lower the channel sampling rate to within a tolerance of ± 250 ppm of the incoming data rate. With the channel sampling rate then within the ± 250 ppm tolerance, the stuff command adds or deletes gated clocks until the output data rate is synchronous with the input data rate. Hence, the coarse rate conversion circuits are independent of the

positive and negative stuff functions performed in the multiplexer. The stuffing of gated clocks does not change the basic bit rate K · Rp of the generated gated clocks where K is the number of used ports strapped together to service an active channel. The addition or deletion of these gated clocks is accomplished as part of the multiplexer's overhead service capability.

Data leaving the multiplexer is placed into a predetermined format as shown in Figure 13. This message format contains data bits sampled from the multiplexer's channel data inputs interleaved with overhead data bits generated within the multiplexer. Overhead data are inserted for two distinct purposes: (1) to provide a framing or synchronization pattern used by the receiving demultiplexer, and (2) to inform the demultiplexer of stuffing actions taken by the multiplexer as part of its input rate buffering capability.

The largest grouping in the output message format is the major frame, which contains 31 complete overhead messages in addition to sampled channel data. Each overhead message consists of stuffing information and a framing pattern corresponding to one of the multiplexer's 31 ports. An overhead word is composed of one complete overhead message in each minor frame. A minor frame contains 29 data words and one complete 29-bit overhead word. This is the second largest organized grouping in the output message format. One bit of the overhead word as shown in Figure 14 is located at the first bit position of each of the 29 data words in the minor frame.

Each data word contains a sampled channel data bit for each port of the multiplexer with the overhead data bit occupying the first data word bit position. The total number of data word bits is variable, a function of the number of multiplexer ports in use. Each multiplexer configuration must use a minimum of 15 ports giving a minimum data word length of 16 bits. This word length can range up to 32 bits when all 32 ports are in use.

When CRC is utilized, the CRC circuits increment a word counter to generate a 10-bit binary code that sequentially advances one count during each word until the end-of-scan signal in word 24 of minor frame count 31 occurs. At this time, the word counter is reset to zero. The end of scan signal together with word 24 clocks a data bit into a 5 bit shift register so an inhibit signal is clocked from the register during word 29. This signal inhibits the word counter and gated clock control so a gated clock cannot be deleted during word 29 which is used for the normal overhead service function. Each incremented word count from the word counter is applied to Input A of a comparator. Before the circuits are activated, the strapping switches on the multiplexer are set to a predetermined value related to the number of gated clocks to be deleted during the major frame. This 10-bit binary code from the strapping switches is applied to Input B of the comparator which compares the 10-bit binary word from the word counter with the 10-bit strapping switch code word. Each time the binary code applied to Input A is greater than Input B, the gated clock inhibit signal is generated. In straight binary logic, the gated clocks to be deleted when A is greater than B would be grouped together. This would cause an undesirable series of consecutively gated clocks. To prevent this occurrence, the word counter
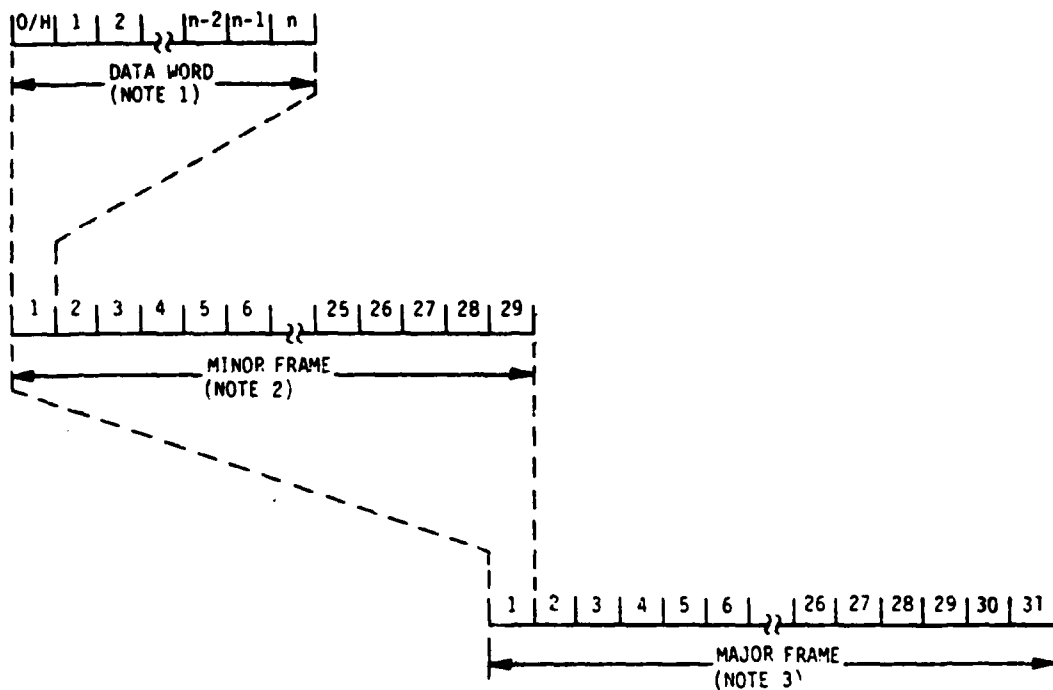
NOTES:

1. EACH DATA WORD CONTAINS ONE OVERHEAD (O/H) BIT PLUS ONE DATA
   BIT FOR EACH USED PORT. TOTAL NUMBER OF WORD BITS FOR A
   GIVEN CONFIGURATION IS BETWEEN 16 and 32.

2. EACH MINOR FRAME CONTAINS 29 DATA WORDS. AN O/H BIT FROM
   EACH DATA WORD MAKES UP ONE 29-BIT O/H WORD PER MINOR FRAME.

3. EACH MAJOR FRAME CONTAINS 31 MINOR FRAMES.

Figure 13. AN/GSC-24 Output Message Format

5-3

22

ONE MINOR FRAME

DATA WORD 1 · DATA WORD 2 · DATA WORD 28 · DATA WORD 29

O/H 1 2 n-2 n-1 n O/H 1 2 n-2 n-1 n | O/H 1 2 3 n-2 n-1 n O/H 1 2 n-3 n-2 n-1 n

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

STUFF CODE (1 of 3) (23 BITS)

PORT NUMBER (5 BITS)

BIT SENSE (1 BIT)

ONE OVERHEAD WORD (29 BITS)
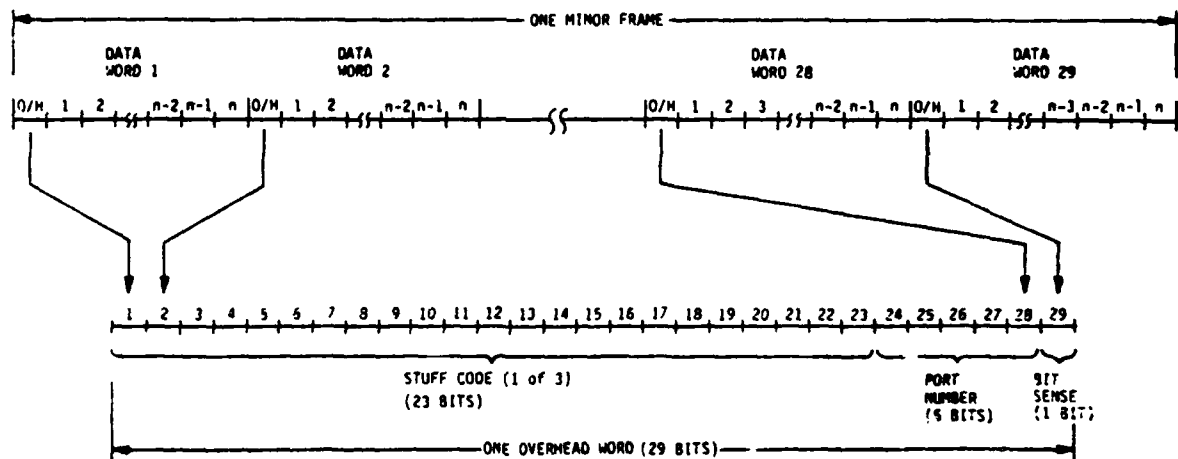
NOTE: VALUE OF n IN DATA WORD IS BETWEEN 15 and 31.

Figure 14. AN/GSC-24 OVERHEAD MESSAGE FORMAT

23

outputs are reversed so the most significant bit (MSB) of the counter is applied to the least significant bit (LSB) of the A input of a 1-bit comparator. This results in a homogeneous spread of the gated clock deletions over the major frame.

In the demultiplexer, the data bits are normally written into an elastic buffer except for the stuffing bits. When the overhead bits identify stuffing bits, the clock is inhibited, causing the stuff bit not to be written into the buffer. If excess data information were carried in these overhead bits, they would be written into the elastic buffer at the correct location of the bit stream. This abrupt discontinuous clocking of the bit stream causes pulse stuffing jitter. The smoothing loop generates the clock to read data from the elastic buffer. This loop contains a phase lock loop which attempts to smooth the abrupt discontinuities and other jitter. It is intended to clock the data stream out of the buffer at a stable rate.

2. SMOOTHING LOOP DESIGN

A schematic of the GSC-24 smoothing loop is given in Figure 15(a). This second order analog phase-locked loop (APLL) consists of a phase detector that produces an error pulse train whose frequency is proportional to the phase error between the input signal and the voltage-controlled multivibrator (VCM) output. The loop filter has a low-pass characteristic which averages the phase error to control the VCM such that the phase error becomes zero. The GSC-24 loop filter is realized with an active network, as shown in Figure 15(b). Two important loop characteristics are the damping factor, n, and the natural frequency of the loop, ω_n, which are related to loop parameters by

$$\omega_n = \left(\frac{k_p k_v}{R_1 C_1 N} \cdot \frac{-j\omega T_3 - 1}{j\omega T_2 + 1} \right)^{1/2} \approx \left(\frac{k_p k_v}{R_1 C_1 N} \right)^{1/2} \tag{20}$$

and

$$n = \frac{R_2 C_1}{2} \cdot \sqrt{\frac{k_p k_v}{R_1 C_1 N}} = \frac{R_2 C_1}{2} \omega_n \tag{21}$$

where

$$T_2 = R_2 C_2$$

$$T_3 = R_2 (C_1 + C_2) \ . \tag{22}$$

24

(a) GSC-24 SMOOTHING LOOP
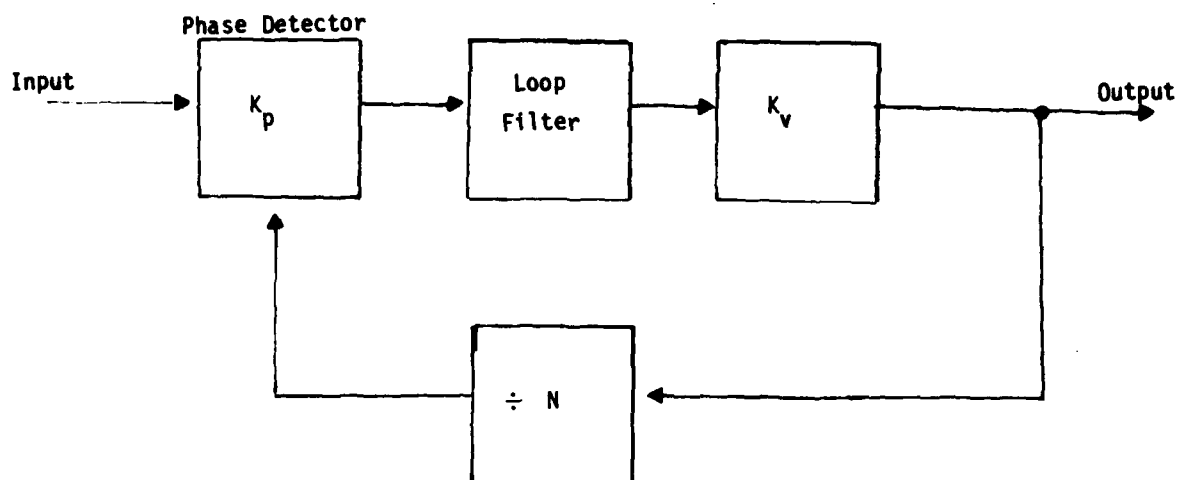
(b) LOOP FILTER SCHEMATIC
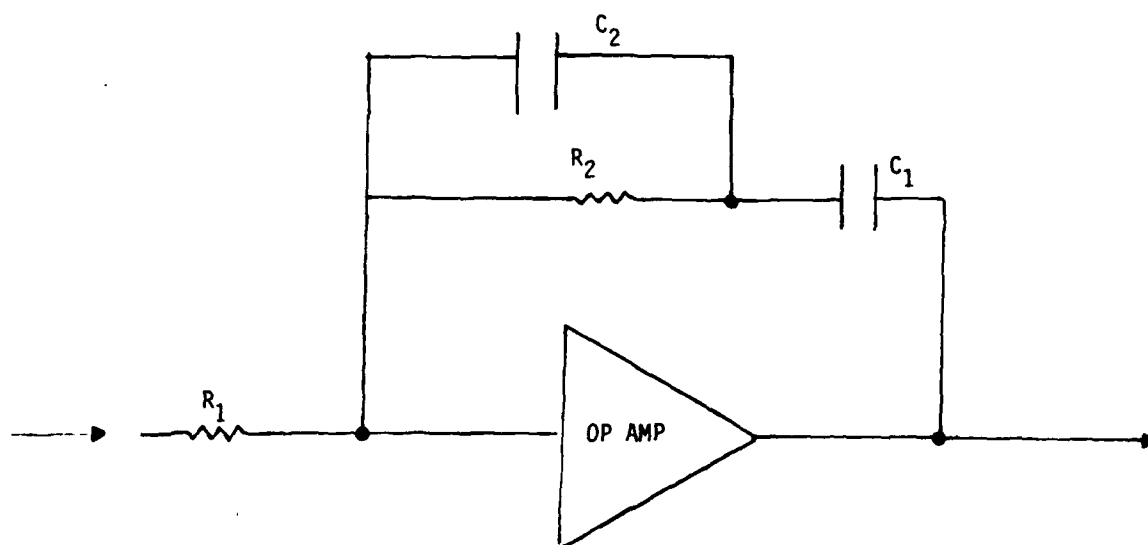
K_p = gain of phase detector
K_v = VCM gain
VCM = Voltage-Controlled Multivibrator

Figure 15. GSC-24 Smoothing Loop and Loop Filter Characteristics

25

Another important loop parameter is the 3-dB bandwidth [4], which is given as a function of the second order loop's damping factor, η, and natural frequency, ω_n:

$$\omega_{3\ dB} = \omega_n \cdot \left[2\eta^2 + 1 + \left\{ (2\eta^2 + 1)^2 + 1 \right\}^{\frac{1}{2}} \right]^{\frac{1}{2}} \tag{23}$$

In applying the smoothing loop to a particular user frequency (bit rate), appropriate choices of loop parameters are made by strapping options in the GSC-24. Within the smoothing buffer (SB) card of the GSC-24, the natural frequency of the loop filter is controlled by switch S5 which allows the selection of the parameter N, while the damping factor of the loop filter is controlled by switch S12, which allows selection of R_2 and C_2. Resulting loop parameter values for selected low speed data rates and recommended switch settings are given in Table I. As part of the AN/GSC-24 testing conducted in August 1980 at the ITF, Ft. Monmouth, NJ, certain experimental switch settings were tried in an attempt to improve interface performance [5]. Table II shows loop parametric values which resulted from optimum switch settings found during this experimentation.* Note that with the S12 switch set to B', the 3 dB bandwidths of this smoothing loop for each data rate were significantly reduced as compared to loop bandwidths for normal S12 settings. As expected, for narrower loop bandwidths, jitter performance showed marked improvement (see Tables IV thru VI).

TABLE I. SMOOTHING LOOP PARAMETRIC VALUES FOR AN/GSC-24 WITH RECOMMENDED SETTINGS (S12 SWITCH = C) ON SMOOTHING BUFFER CARD

Bit Rate (b/s)	K_p	K_v	R_1 (M ohms)	C_1 (μF)	N	R_2 (k ohms)	ω_n (Hz)	η	ω_{3dB}
9600	.111	27650	3.3	22	32	270	1.1494	3.414	9.886
4800	.111	27650	3.3	22	64	270	.8127	2.414	4.092
2400	.111	27650	3.3	22	128	270	.5747	1.707	2.129
1200	.111	27650	3.3	22	256	270	.4064	1.207	1.146

*Discussions with Martin Marietta engineers indicated that choice of non-standard switch settings may result in GSC-24 instability, so that this approach for improving performance was discarded.

TABLE II. SMOOTHING LOOP PARAMETRIC VALUES FOR AN/GSC-24
WITH EXPERIMENTAL SETTINGS (S12 SWITCH = B')
ON SMOOTHING BUFFER CARD

Bit Rate (b/s)	K_p	K_v	R_1 (M ohms)	C_1 (μF)	N	R_2 (k ohms)	W_n (Hz)	η	ω_{3dB} (rads/sec)
9600	.111	27650	3.3	22	32	18.2	1.1494	0.2301	1.8522
4800	.111	27650	3.3	22	64	18.2	.8127	0.1627	1.2863
2400	.111	27650	3.3	22	128	18.2	.5747	0.1151	0.9013
1200	.111	27650	3.3	22	256	18.2	.4064	0.0814	0.6344

3. INTERFACE TEST RESULTS

A brief description of past GSC-24 interface testing is useful in
understanding the problem of slewing jitter and its effect on other
equipment. The phase I (R&D) model of GSC-24 employed a smoothing buffer
based on a digital first order loop. A measured slew rate of 2π in 768 bit
times resulted [6], but since several sink equipments could not accept this
slew rate, modifications were made to extend the smoothing interval. However,
this modification to the digital smoothing loop also restricted the allowed
data source rate offset. For Phase II (production) units, the GSC-24 was
reconfigured and partially redesigned. The reconfiguration consisted
primarily of a reduction in the number of channels (user inputs) from 31 to
15, and the redesign consisted primarily of a change from a digital first
order loop to an analog second order loop for demultiplexer output smoothing.
Figure 16 shows the slew rates determined by calculation from the second order
analog loop equations. In practice, it is desirable to measure the actual
slew rate produced by a given smoothing buffer. One successful technique uses
a delayed trigger oscilloscope monitoring the output timing. One bit, several
bit times after the trigger point, is displayed and the peak bit-to-bit jitter
is measured. This jitter represents the phase change that has occurred since
the trigger point. If no stuffing action has occurred, the bit edge remains
motionless except for very small movement from random bit-to-bit jitter and
noise. The maximum amount of jitter is seen when the sampling interval on the
scope occurs during the steepest slope section of the smoothing action (see
Figure 9). A worst case approximation of the slew rate can be found by

27
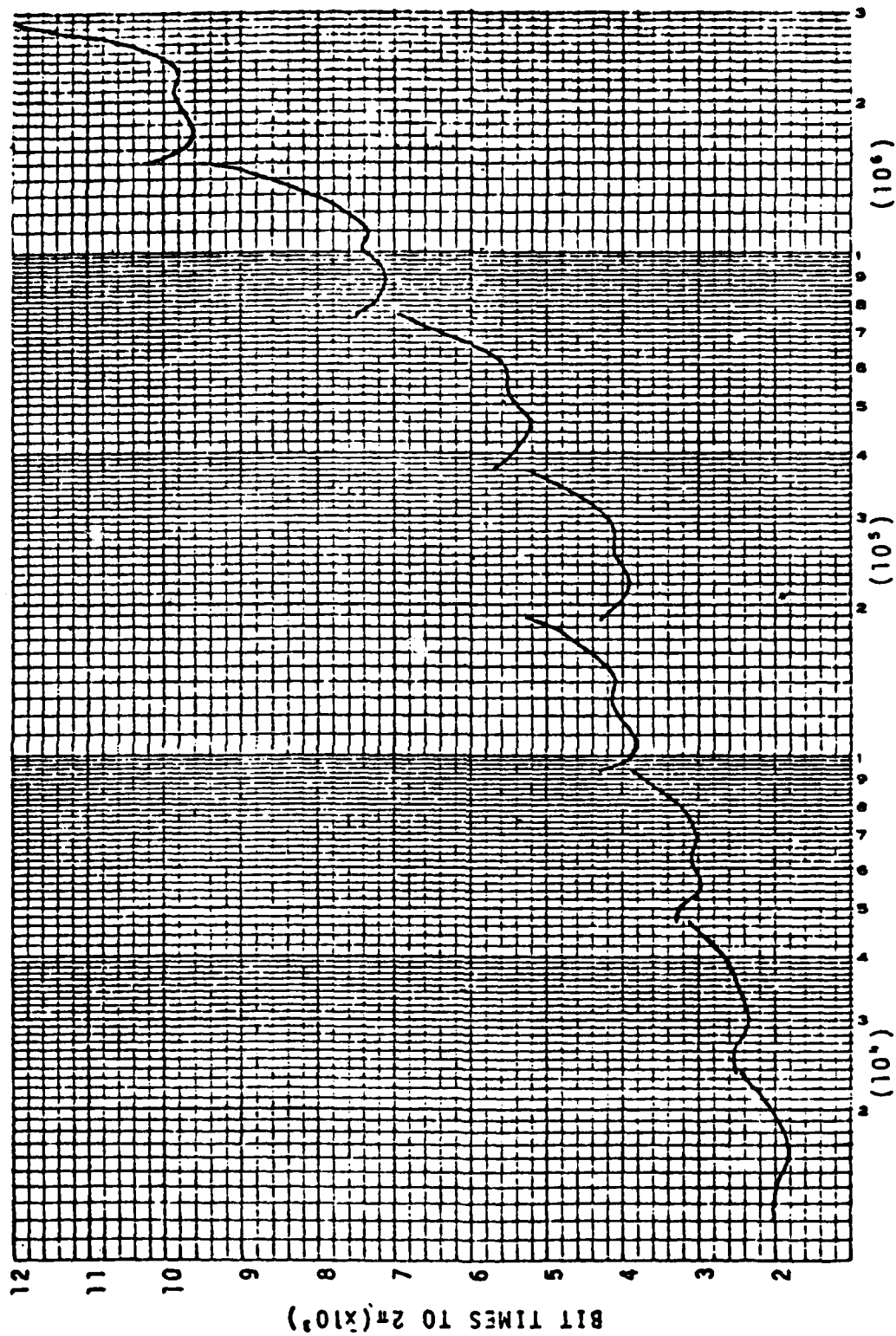
Figure 16. Slew Rate vs. Data Rate (BPS)

BIT TIMES TO 2π (×10^3)

extrapolation of the steepest slope section of the smoothing action. As an example, if the phase change (peak-to-peak jitter) were measured as 5% over a 200 bit interval, then a 100Δ (2π) change would occur in 4000 bits, giving a slew rate of 2π in 4000 bit times.

a. HN-74 Interface Test. The April-May 1979 compatibility test of various GFE equipment, including the MD-921/G PSK Modem, CV-3034 A/D Converter, AN/USC-26 Group Data Modem, MD-823/G Channel Modem, and HN-74 Interface Device, has indicated various degrees of difficulty in interfacing the GSC-24 [6]. As an example, interface tests with the HN-74 operating at its standard rate of 1.544 Mb/s indicated that the HN-74 could not track the frequency variations output by the GSC-24 [7]. At 1.544 mb/s the GSC-24 slew rate is between 6 and 10 K bit times for a 2π phase change (see Figure 16). At low offsets where stuffs occur less frequently than every 64 bits (below + 250 Hz) the loop tracks out each stuff with a frequency offset, then returns back to 1.544 Mb/s. The average offset during the smoothing interval is roughly 150-225 Hz with peak offsets to 450 Hz. Since the HN-74 can only accept offsets of 150-250 Hz, the bit synchronizer in the HN-74 occasionally loses phase lock, resulting in loss of bit count integrity. This problem has been partially solved by the use of coarse rate conversion (CRC) within the GSC-24. One particular test configuration utilized 100 fill bits per major frame, giving a channel rate of 1.566 Mb/s in the multiplexer. With this increased channel rate, additional stuffing actions occur during the nominal 6-10 kilobit smoothing interval. As a result the loop no longer tracks individual stuffs but now goes to the average channel rate and makes small variations around it. On the average, a stuff action occurs every 70 bit times, allowing a very smooth output. Configuring the GSC-24 channel in the coarse rate mode allowed operation with the HN-74 over the range + 70 Hz without degradation.

b. VF Modem Interface Tests. More recently, interface compatibility tests of the GSC-24 with several synchronous voice channel modems have indicated unsatisfactory performance. In May 1979, DCEC issued a requirement for GSC-24 and voice channel modem integration testing [5], which was subsequently conducted at the integrated test facility (ITF), SATCOMA, Ft. Monmouth, NJ. The effects of pulse stuffing and course rate conversion were to be investigated. The multiplex configurations used were plans F and G from the DCEC Test plan [5], as shown here in Table III. In all cases, a RS-232/MIL-STD-188 interface was used between the modems and GSC-24. The test configuration used for each modem is shown in Figure 17. Modems tested were the (1) CODEX 9600/V.29, (2) RACAL MILGO MPS 9601, and (3) WECO 9600.

For the initial set of tests, the modem clock was not synchronized to the GSC-24 reference clocks, since this would be a normal operational configuration. Also, normal strap settings on the SB card were used, namely, strap C for switch S12 and strap C for switch S5. As shown in Figure 17, a bit error rate test set, the TS 3642, was used to determine the BER for each modem/GSC-24 configuration. Configuration details and BER performance are given in Tables IV-VI for the three modems. In all cases, unsatisfactory performance was observed for the three modems using multiplex plan F (without fill bits). The cause of unsatisfactory performance was a loss of modem synchronization due to GSC-24 pulse stuffing jitter. This loss of modem

synchronization was evident from bursts of errors, and correlation of loss of synchronization with pulse stuffing events was evident through waveform observation on an oscilloscope. In an attempt to improve this observed performance, other non-standard strap settings on the SB board were made and BER tests conducted with results as shown in Tables IV-VI. The strap setting on switch S12 was changed from C, which is recommended for rates of 1.2 to 9.6 kb/s, to B and B', which are recommended for higher rates, in order to lengthen the smoothing interval from 1000 bit times to 2500 bit times. In so doing, performance at 7200 and 9600 b/s was improved, with, in fact, a zero BER for the B' settings for all rate offsets tested and for all three modems. Also for the rate 4800 b/s with switch S12 again set with a B' strap, performance likewise improved with zero BER readings, although it was necessary to strap the S5 switch to the 6-12 kb/s position in order to obtain a zero error rate for the CODEX LSI 9600/V.29 modem.

For multiplex plan G, which used fill bits, the error rate did not appear to increase over the values shown in Tables IV-VI, although errors still occurred coincident with pulse stuffing. When a buffer was interposed between the demultiplex channel output and the modem data input, as shown in Figure 18, error free operation was observed for an interval consistent with the clock rate difference between the test set (reading into the buffer) and the modem (reading out of the buffer) clocks. The buffer compensated for the phase slewing action associated with both stuff and fill bit deletion in the demultiplexer. However, in using the internal clocks of the test set and modem, the interval between buffer resets proved insufficient to provide satisfactory performance. A cesium beam standard with two frequency synthesizers was then introduced to provide highly accurate and stable clock frequencies to both the test set and modem. As expected, buffer reset periods were extended to operationally useful intervals for all modems under test.

Conclusions reached on the basis of these observations are:

(1) Pulse stuffing jitter on the GSC-24 demultiplexer clock signal causes frequent loss of synchronization in each modem's clock recovery circuitry.

(2) For certain non-standard strap settings within the smoothing buffer (SB) card of the GSC-24, acceptable performance was realized with each modem for most bit rates. This performance resulted from an extension of the smoothing interval sufficient to allow tracking by modem bit synchronizers. However, use of these non-standard strap settings is not recommended because of unknown performance for all sets of configuratons, especially for tandem or nested operation. Martin-Marietta, the GSC-24 contractor, was consulted regarding this fix to the problem and advised against changing the standard strap settings.

(3) Use of a buffer and a high quality station clock provided satisfactory performance, but represents a high cost solution.

(4) Approaches other than those described above have been selected for interface of VF modems, including (a) use of VF analog channels and (b) use of synchronous multiplexing, such as the LSTDM.

Table III. AN/GSC-24 Multiplex Plan

```
***********************************************************************
* XTAL = 0              * TDM STAGE 1 *   PORT RATE=    1.200 KBPS   *
***********************************************************************
* CH PORTS   TYPE   CKT RATE PILL   STRAPPING        REMARKS         *
* --------   ----   -------- ----   ---------        -------         *
*   1    8   RCB/SB   9.600    0    0000000000                       *
*   2    6   RCB/SB   7.200    0    0000000000                       *
*   3    4   RCB/SB   4.800    0    0000000000                       *
*   4    2   RCB/SB   2.400    0    0000000000                       *
*   5    1   RCB/SB   1.200    0    0000000000                       *
*   6    .      .        .     .                                     *
*   7    .      .        .     .                                     *
*   8    .      .        .     .                                     *
*   9    .      .        .     .                                     *
*  10    .      .        .     .                                     *
*  11    .      .        .     .                                     *
*  12    .      .        .     .                                     *
*  13    .      .        .     .                                     *
*  14    .      .        .     .                                     *
*  15    .      .        .     .                                     *
*        1   OVERHEAD      .      .                                  *
***********-----*****************************************************
*       22 PORTS                    TDM XMT RATE=    26.400 KBPS     *
***********************************************************************
```

LINK TRANSMISSION RATE = 26.400 KBPS
CRYSTAL 1 ON ALL STAGES

(A) AN/GSC-24 MULTIPLEX PLAN F

```
***********************************************************************
* XTAL = 0              * TDM STAGE 1 *   PORT RATE=    2.315 KBPS   *
***********************************************************************
* CH PORTS   TYPE   CKT RATE PILL   STRAPPING        REMARKS         *
* --------   ----   -------- ----   ---------        -------         *
*   1    5   RCB/SB   9.600   767   1110000111                       *
*   2    4   RCB/SB   7.200   800   0111010111                       *
*   3    3   RCB/SB   4.800   833   1010101111                       *
*   4    2   RCB/SB   2.400   866   1011111111                       *
*   5    1   RCB/SB   1.200   433   0111111110                       *
*   6    .      .        .     .                                     *
*   7    .      .        .     .                                     *
*   8    .      .        .     .                                     *
*   9    .      .        .     .                                     *
*  10    .      .        .     .                                     *
*  11    .      .        .     .                                     *
*  12    .      .        .     .                                     *
*  13    .      .        .     .                                     *
*  14    .      .        .     .                                     *
*  15    .      .        .     .                                     *
*        1   OVERHEAD      .      .                                  *
***********-----****************************************************
*       16 PORTS                    TDM XMT RATE=    37.040 KBPS     *
***********************************************************************
```

LINK TRANSMISSION RATE = 37.040 KBPS
CRYSTAL 1 ON ALL STAGES

(B) AN/GSC-24 MULTIPLEX PLAN G
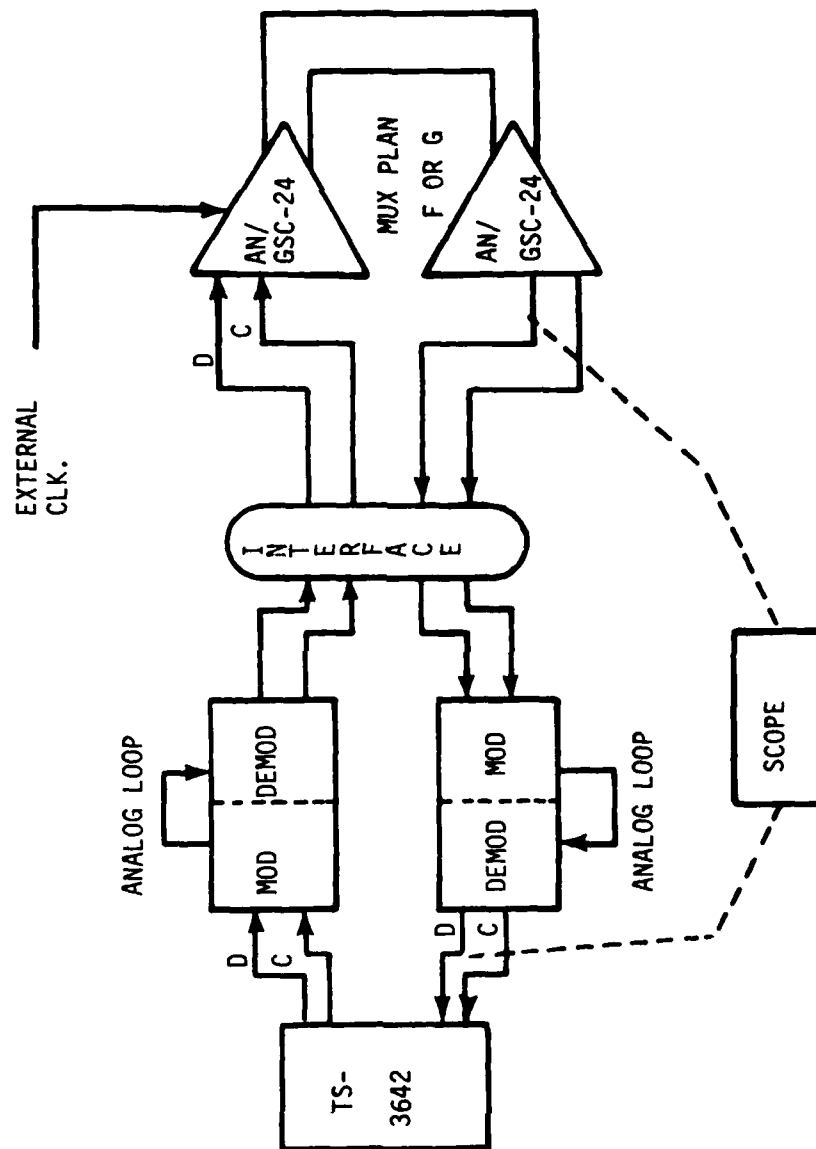
Figure 17. AN/GSC-24 Interface Tests With VF Modems

32
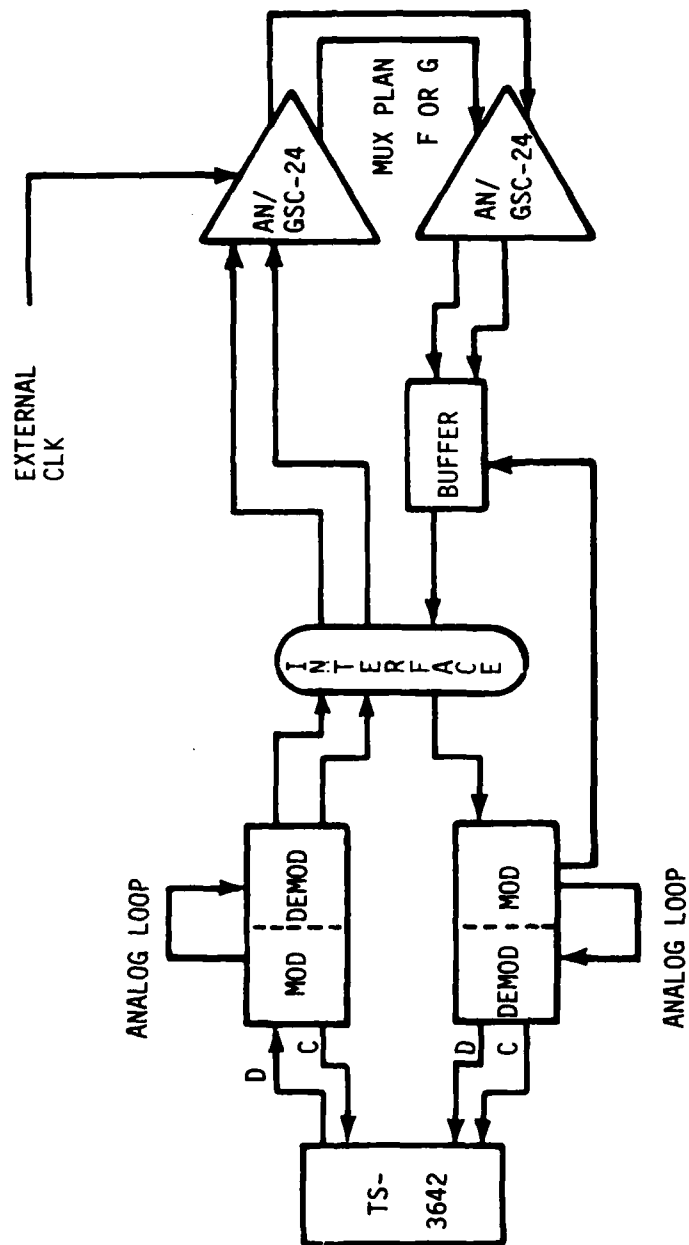
Figure 18. AN/GSC-24 Interface Test With VF Modems Using Buffers

33

TABLE IV.   CODEX LSI 96/V.29 TEST WITH AN/GSC-24
FOR MULTIPLEX PLAN F.

Channel	Bit Rate (b/s)	Strap S12	Strap S5	Block Length (bits)	Rate Offset (BPS)	Error Rate
1	9600	C	6–12k	10^7	0	8.5×10^{-6}
		B	6–12k	10^6	± 1.0	4.0×10^{-5}
		B'	6–12k	10^6	± 2.0	zero
2	7200	C	6–12k	10^6	$+.1$	4.3×10^{-5}
		B'	6–12k	4×10^6	$+1.4$	zero
		B'	6–12k	10^6	-1.4	zero
3	4800	C	3–6k	10^6	$+.1$	1.0×10^{-4}
		B'	3–6k	10^6	$+.1$	2.3×10^{-5}
		B'	6–12K	2.7×10^8	$+.1$	zero
		B	6–12k	10^6	$-.1$	5×10^{-6}

TABLE V. RACAL MILGO 9601 TEST WITH AN/GSC-24
FOR MULTIPLEX PLAN F

Channel	Bit Rate (b/s)	Strap S12	Strap S5	Block Length (bits)	Rate Offset (BPS)	Error Rate
1	9600	C	6–12k	10^6	0	6×10^{-6}
		B'	6–12k	10^6	+1.1	zero
		B'	6–12k	10^6	−0.9	zero
2	7200	C	6–12k	10^6	0	5×10^{-6}
		B'	6–12k	10^6	+1.4	zero
3	4800	C	3–6k	10^6	0	5×10^{-4}
		B'	3–6k	10^6	+0.5	zero
4	2400	C	1.5–3k	10^6	0	5.7×10^{-5}
		B'	1.5–3k	10^6	+0.3	3.0×10^{-3}

TABLE VI. WECO 9600 TEST WITH AN/GSC-24 USING MULTIPLEX PLAN F

Channel	Bit Rate (b/s)	Strap S12	Strap S5	Block Length (bits)	Rate Offset (BPS)	Error Rate
1	9600	C	6-12k	10^6	0	2.0×10^{-3}
		B	6-12k	10^6	0	zero
		B	6-12k	10^6	+1.0	6.1×10^{-4}
		B'	6-12k	10^6	± 1.0	zero
2	7200	C		10^6	0	2.2×10^{-3}
		B		10^6	+0.7	5×10^{-6}
		B'		10^6	± 0.7	zero

IV. DESCRIPTION OF PULSE STUFFING SIMULATION

The purpose of this simulation is to compute the steady state performance of a pulse stuffing multiplexer. The system to be simulated consists of a number of multiplexer/demultiplexer pairs, referred to as nodes, cascaded in series as would typically be done in a network implementation. A basic criterion of the software simulation is the capability to take the jittered output of a node and make that the input to another node to permit the simulation of cascaded nodes.

Figure 19 is a pictorial representation of the system model showing two nodes in cascade followed by a bit synchronizer. Each node has four frequencies of interest. These are:

a. The input frequency (f_1) which is a T1 channel tributary source for the first node.

b. The transport frequency (f_2) for synchronous transmission between nodes, called the T2 rate frequency.

c. The nominal frequency of the VCO (f_3) used in the smoothing loop for that node.

d. The output frequency which is the input frequency to the following node.

The parametric studies of the jitter cascade effect requires that the simulation consist of first or second order phaselock loops at each node followed by a second phaselock loop representing the bit synchronizer.

The operation of the elastic buffer in conjunction with overhead bits to indicate a stuffing condition is modeled as a phase detector which compares the input frequency (f_1) with the node data transport frequency (f_2) and makes discrete phase corrections to the f_2 input. This corrected f_2 clock rate is an input to a phase-locked loop which models the smoothing that occurs in the system for reading data out of the elastic buffer. The phase error sampler is driven by the T2 rate. The frequency out of the first node is a jittered T1 rate which is an input to node 2. For the general case, the data stream continues through n cascaded nodes and terminates with a model of a bit synchronizer. The bit synchronizer is another phase-locked loop; the difference is that the parameter of interest for the bit synchronizer is the phase error (the output of the phase-locked loop's summing node or the input to the sampler) which determines total system error in detecting the proper bit in the data stream.

Each node contains a digital phase-locked loop to simulate the demultiplexer smoothing loop. The last node includes simulation of a bit synchronizer (receiver) as the final component. This node model is a linear model which is valid for steady-state performance analysis. Since study of the steady-state performance is the desired simulation output, the nonlinear performance of the phase comparator during the acquisition phase can be ignored. The system is then subjected to repeated transient impacts due to

37

pulse stuffing and fill bit insertions. The natural simulation model implementation for a digital phase lock loop is a discrete simulation with discrete time inputs. However, this system is highly oversampled since the phase-locked loop bandwidth is very low compared to the sample rate. This condition permits the simulation model of the phase locked loop to be implemented as a continuous system defined via differential equations to provide faster solutions on the computer with identical dynamic response. The variable step size Runga-Kutta integration algorithm permits relatively large time increments for integration after the transient from each pulse stuffing or fill bit insertion activity.

The dynamic equations are table-driven to permit generalized routines for first or second order nodes that are accessed based on the number of nodes being simulated for any one simulation run. Discrete events include the opportunity to advance or retard the node phase and the periodic collection of phase and frequency statistics for analysis and plotting. Minimum and maximum phase peaks are detected which trigger a state-event for collection of statistics on peak phase variation.

To represent each hardware module set (node), two items are required:

(1) A discrete-time simulation of the periodic phase correction opportunities with a decision criteria to determine the phase correction time.

(2) A continuous (either state equation or difference equation) simulation of the smoothing loop's effect on the phase relationship of the jittered signal passing through the smoothing loop with respect to the stable input signal.

Since we are modeling a linear system (with additive elements), a perturbational model of phase jitter is the preferred simulation approach to guarantee computational accuracy. This is accomplish by setting the T1 rate input frequency equal to zero.

The cascade effect is modeled by using the first node's output as the second node's input and repeating for the number of simulated nodes. Figure 20 is a pictoral representation of the perturbational model showing two cascaded nodes.

To complete the overall model, it is necessary to model the bit synchronizer as the final element of these cascaded nodes. The bit synchronizer is a phase-locked loop whose parameter of interest is the phase error, i.e., the output of the phase-locked loop's summing node.

The simulation objectives are to obtain answers to the pulse stuffing, fill bit problems of various multiplexers as functons of multiple configurations, input rates, output rate offsets, and internal strapping options in smoothing loops. This is accomplished by measuring peak jitter, rms jitter, and frequency slew rate versus time from plots generated by the GASP simulation program. These plots show various phase locked loop parameters plotted as a function of time.

38
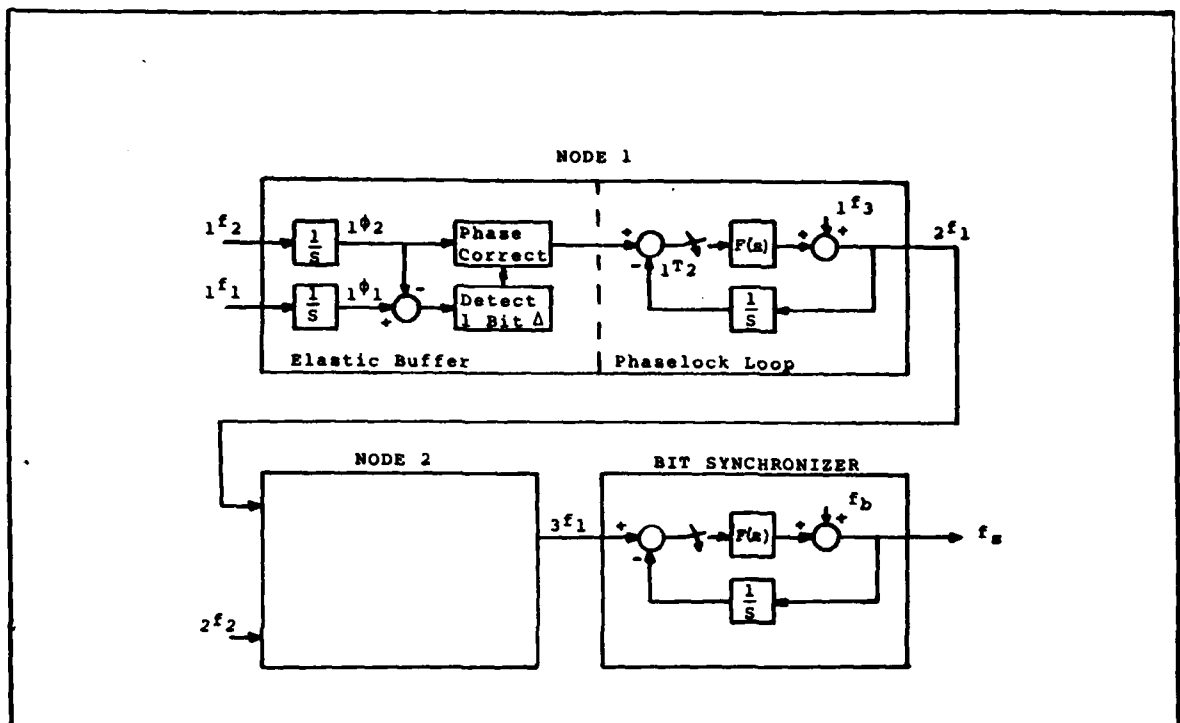
Figure 19.   System Model
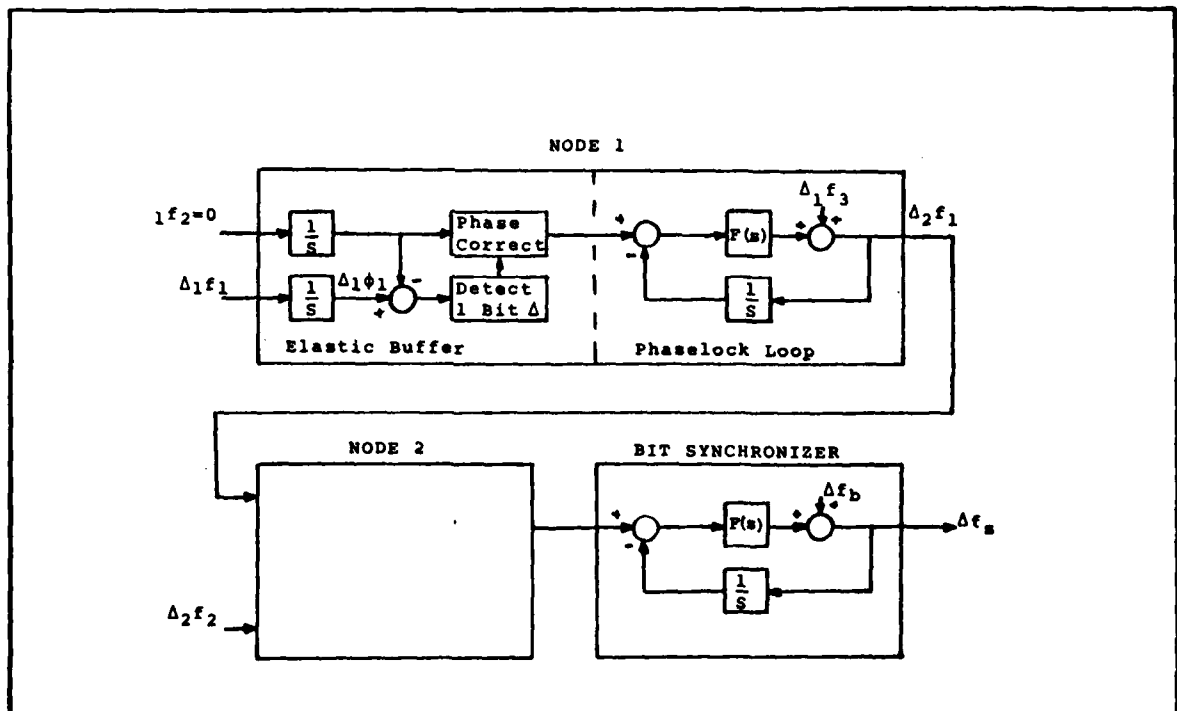
Figure 20.   System Perturbation Model

39

V. RESULTS AND CONCLUSIONS FROM AN/GSC-24 SIMULATION

The simulated configurations were extracted from multiplex plans found in reference [5]. Actual hardware tests were conducted in August 1979 at the Integrated Test Facility (ITF), Ft. Monmouth, NJ, as reported in Chapter III, Section 3b of this report. Specifically, multiplex plans F and G of reference [5] were selected for simulation. A comparison of simulation results with actual results could then be accomplished.

For several parametric variations, the demultiplexer's smoothing loop performance has been characterized by simulation. This characterization took the form of (1) peak-to-peak jitter measured from the time response waveforms output from the smoothing loop or (2) slew rate defined as the maximum rate of phase change and measured from the loop output waveforms as a $\Delta\phi$ degree (or radian) phase change in a ΔT number of bit times (or seconds). Since a $90°$ phase change would result in significantly degraded performance, the measurement of slew rate was based on the minimum number of bit times observed for a $90°$ phase change. To facilitate comparison with the test results and other specifications, the resulting slew rate numbers have been extrapolated to 2π radians by multiplying the number of bit times measured times four.

Several computer runs were completed, mostly at channel rates of 4800 and 9600 b/s, to comply with the selected multiplex configurations. For a given computer run, depending on the parameter being varied, output data corresponding to a fraction of a second up to several seconds were examined. Plots of smoothing loop phase error versus time were used in determining peak-to-peak jitter and slew rate. Results of these measurements for the parameters under study are presented below.

1. EFFECT OF STUFF VERSUS FILL BITS

Table VII indicates results of testing multiplex plan F (with no fill bits) versus multiplex plan G (with fill bits) for rates of 4800 and 9600 b/s. Nominal (recommended in GSC-24 Operations Manual [8] parametric values were used for the smoothing loop bandwidth and damping factor. Note that multiplex configurations which employed fill bits (for rate conversion) were found to have better slew rate characteristics. Intuitively, a larger number of fill or stuff bits should result in smaller phase changes required to keep the smoothing loop output frequency about some average (and desired) value. A fewer number of fill or stuff bits would result in larger phase transients about the desired frequency. Given this intuitive reasoning, slew rate performance should improve as the input frequency is offset by greater amounts, since larger offsets will result in a higher stuffing ratio for a positive-negative pulse stuffing multiplexer. Examination of Figure 22 shows this effect, where improved slew rate performance is observed for ±250 ppm offsets as compared to ±10 ppm offsets for up to four nodes in cascade.

2. EFFECT OF POSITIVE VERSUS NEGATIVE STUFFING

Table VIII gives results of several simulation runs aimed at showing the effects of positive versus negative pulse stuffing on the slew rate characteristic. These data were collected by offsetting the input frequency by equal amounts but in opposite directions. For a positive offset, the GSC-24 will use only positive stuffing; for negative offsets, only negative stuffing will result. These two effects are due to the positive-negative nature of GSC-24 pulse stuffing. Since the smoothing loop is subject to the same absolute magnitude of phase error for equal but opposite offsets, the slew rate characteristic should be unaffected by the sign of the offsets. Examination of Table VIII indicates that indeed the slew rate is not materially affected by change in the sign of the offset.

3. EFFECT OF DAMPING FACTOR

Table IX shows the effect of changing the damping factor in the GSC-24 smoothing loop on the slew rate characteristic. Earlier tests of the GSC-24 interfaced with VF modems, conducted at the ITF, Ft. Monmouth, NJ, indicated improved performance with judicious choice of damping factor. Specifically, strap S12 of the smoothing loop board controlled the damping factor. Use of the B' strap resulted in pronounced improvement as shown in Tables IV through VI. Likewise, simulation results shown in Table IX indicate significant improvement with a choice of the B' strap.

These data also illuminate the relationship between the GSC-24 slew rate and typical VF modem bit synchronizers. It is known that VF modems can accept typically up to $\pm\pi$ radian phase change in 10,000 bit times. A look at Table IX shows that in every case a choice of the C strap for damping factor results in an excessive slew rate. Tables IV through VI reflect this same conclusion, since all modems tested with the C strap resulted in a non-zero bit error rate. Conversely, a choice of the B' strap resulted in acceptable slew rate, as shown in Table IX and as indicated by the zero error rates of Tables IV through VI.

4. EFFECT OF TANDEMING

Figures 21 and 22 illustrate the effect on jitter performance of tandeming GSC-24 multiplexer/demultiplexer pairs (nodes). For this illustration, a 9600 b/s data rate is used from multiplex plan G (see Table III), which was selected so that both fill and stuff bits would be utilized. Figure 21 plots peak-to-peak jitter versus tandemed nodes for up to six nodes, while Figure 22 plots slew rate versus tandemed nodes for up to eight nodes. In obtaining these data, the actual data rate used was offset from 9600 b/s by up to 250 ppm. The curves labeled $\pm$250 ppm resulted from offsetting each nodal clock (f_2, the transport frequency -- see Figure 19) by $\pm$250 ppm. Likewise, those curves labeled $\pm$10 ppm or $\pm$250 ppm resulted from alternating the sign of the offset with each succeeding node. In this way, the effects of rate offsets, in both magnitude and sign, can be observed with tandemed nodes. First note that slew rate and jitter performance degrades with increased tandeming for 250 ppm offsets, regardless of sign. Results for 10 ppm offsets actually show

41

slight improvement in slew rate performance with increased tandeming. These results indicate that jitter and slew rate performance is not strongly dependent on the number of tandemed nodes. This same weak dependence on tandeming has been observed with testing of other pulse stuffing TDM equipment [9].

5. EFFECT ON BIT SYNCHRONIZERS

Curves showing the interaction of a bit synchronizer with a jittered input signal are shown in Figures 23 and 24. These curves show peak-to-peak phase error and maximum slew rate measured at the output of the bit synchronizer for the GSC-24 second order smoothing loop interfaced with a second order bit synchronizer. For these curves a nominal input frequency of 9600 Hz with an offset of 0.1Δ was utilized. The two figures present the information for a constant ratio of bit synchronizer loop bandwidth to smoothing filter loop bandwidth, with the ratio denoted by k. These curves show that jitter performance is improved by increasing the loop bandwidth of the bit synchronizer. For example, from Figure 23, when the smoothing loop has a bandwidth equal to one-tenth the stuffing rate, the phase error ranges from 150° for k=0.093 down to 4.5° for k=0.93.

TABLE VII. EFFECTS OF STUFF VS. FILL BITS ON SLEW RATE

Test Run	Bit Rate	Mux[5] Plan	Fill Bits/ Major Frame	Damping Factor Strap (S12)	Loop Band- width (Hz)	Slew Rate
1	4800	F	0	C	25.7	2π rad/ 1450 bits
2	4800	G	833	C	25.7	2π rad/ 7600 bits
3	9600	F	0	C	62.1	2π rad/ 3200 bits
4.	9600	G	767	C	62.1	2π rad/ 6250 bits

TABLE VIII. EFFECT OF POSITIVE VERSUS NEGATIVE STUFFING ON SLEW RATE

Test Run	Bit Rate	Mux [5] Plan	Fill Bits/ Major Frame	Damping Factor	Positive or Negative Stuff	Slew Rate
2	4800	G	833	C	+	2π rad/ 7600 bits
9	4800	G	833	C	−	2π rad/ 7600 bits
6	4800	G	833	B'	+	2π rad/ 33700 bits
10	4800	G	833	B'	−	2π rad/ 32500 bits
1	4800	F	0	C	+	2π rad/ 1450 bits
11	4800	F	0	C	−	2π rad/ 1460 bits
5	4800	F	0	B'	+	2π rad/ 12200 bits
12	4800	F	0	B'	−	2π rad/ 11700 bits

TABLE IX. EFFECTS OF DAMPING FACTOR ON SLEW RATE

Test Run	Bit Rate	Mux [5] Plan	Fill Bits/ Major Frame	Damping Factor Strap (S12)	Loop Band- width (Hz)	Slew Rate
1	4800	F	0	C	25.7	2π rad/ 1450 bits
5	4800	F	0	B'	8.1	2π rad/ 12200 bits
2	4800	G	833	C	25.7	2π rad/ 7600 bits
6	4800	G	833	B'	8.1	2π rad/ 33700 bits
3	9600	F	0	C	62.1	2π rad/ 3200 bits
7	9600	F	0	B'	11.6	2π rad/ 86500 bits
4	9600	G	767	C	62.1	2π rad/ 6250 bits
8	9600	G	767	B'	11.6	2π rad/ 53760 bits
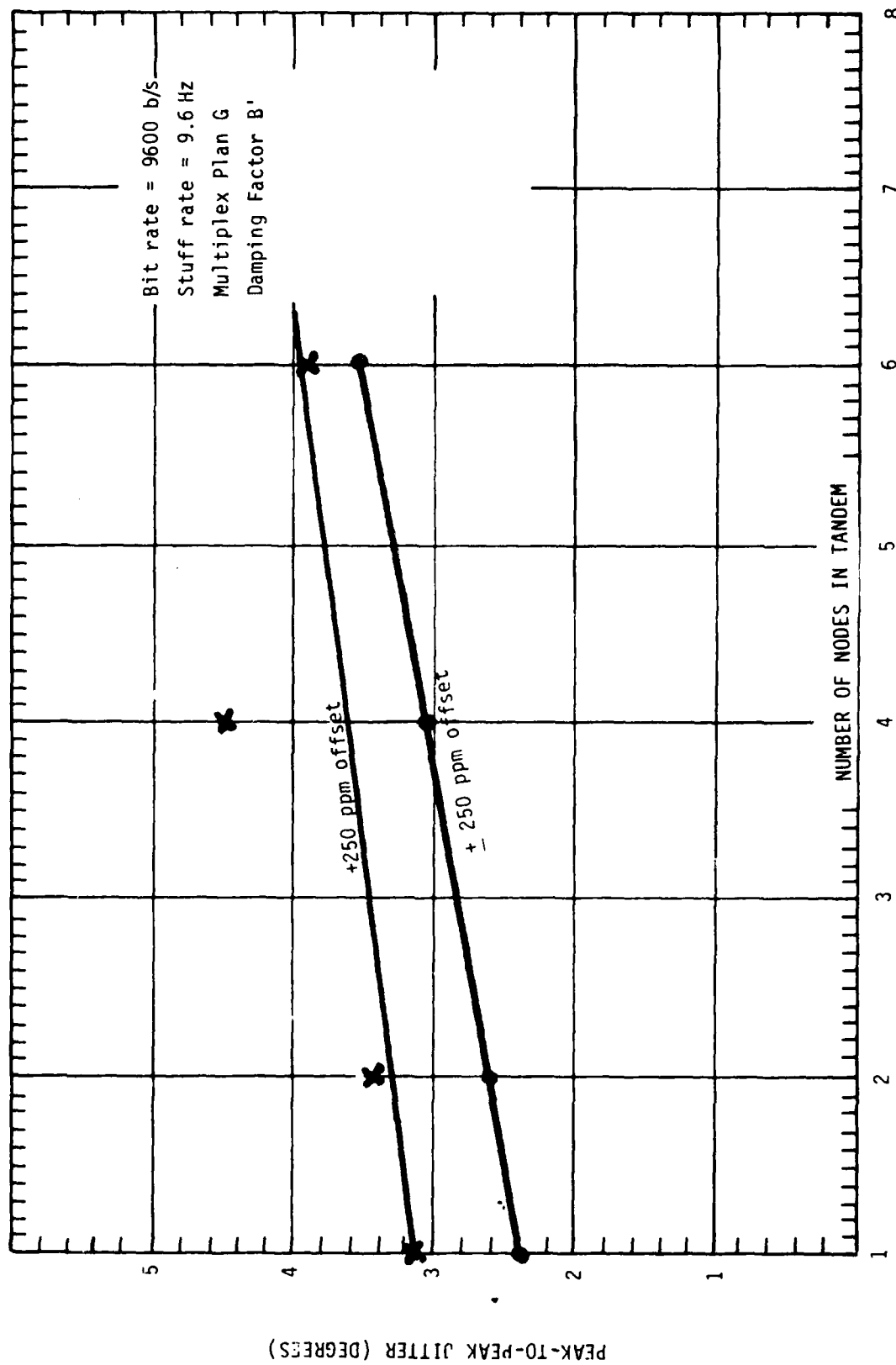
Figure 21. Effecting of Tandeming on Pulse Stuffing Jitter for AN/GSC-24

44

FIGURE 22. EFFECTING OF TANDEMING ON PULSE STUFFING SLEW RATE FOR AN/GSC-24.
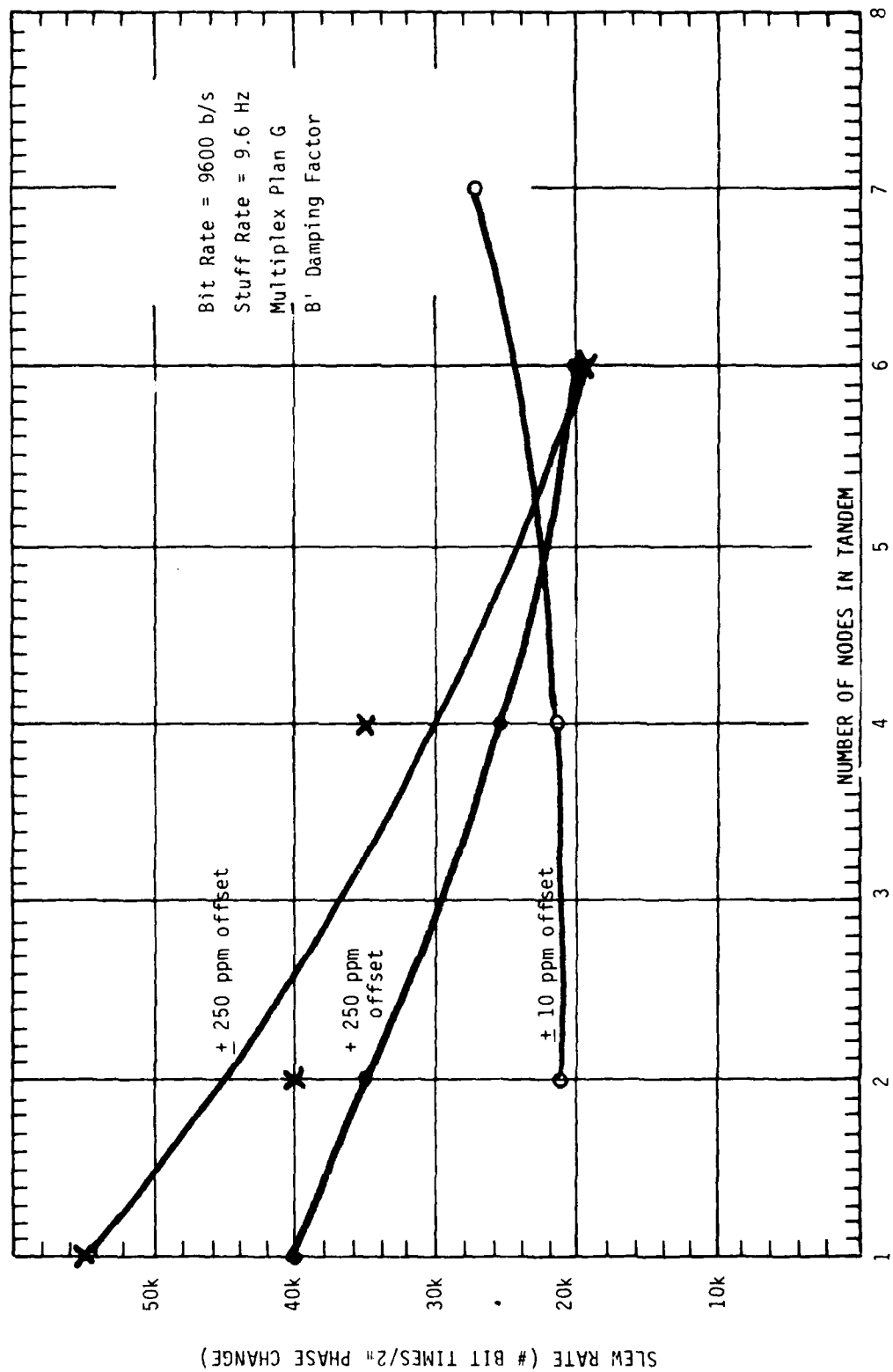
Bit Rate = 9600 b/s
Stuff Rate = 9.6 Hz
Multiplex Plan G
B' Damping Factor

± 250 ppm offset

± 250 ppm offset

± 10 ppm offset

NUMBER OF NODES IN TANDEM

SLEW RATE (# BIT TIMES/2" PHASE CHANGE)

Figure 22. Effecting of Tandeming on Pulse Stuffing Slew Rate for AN/GSC-24

45
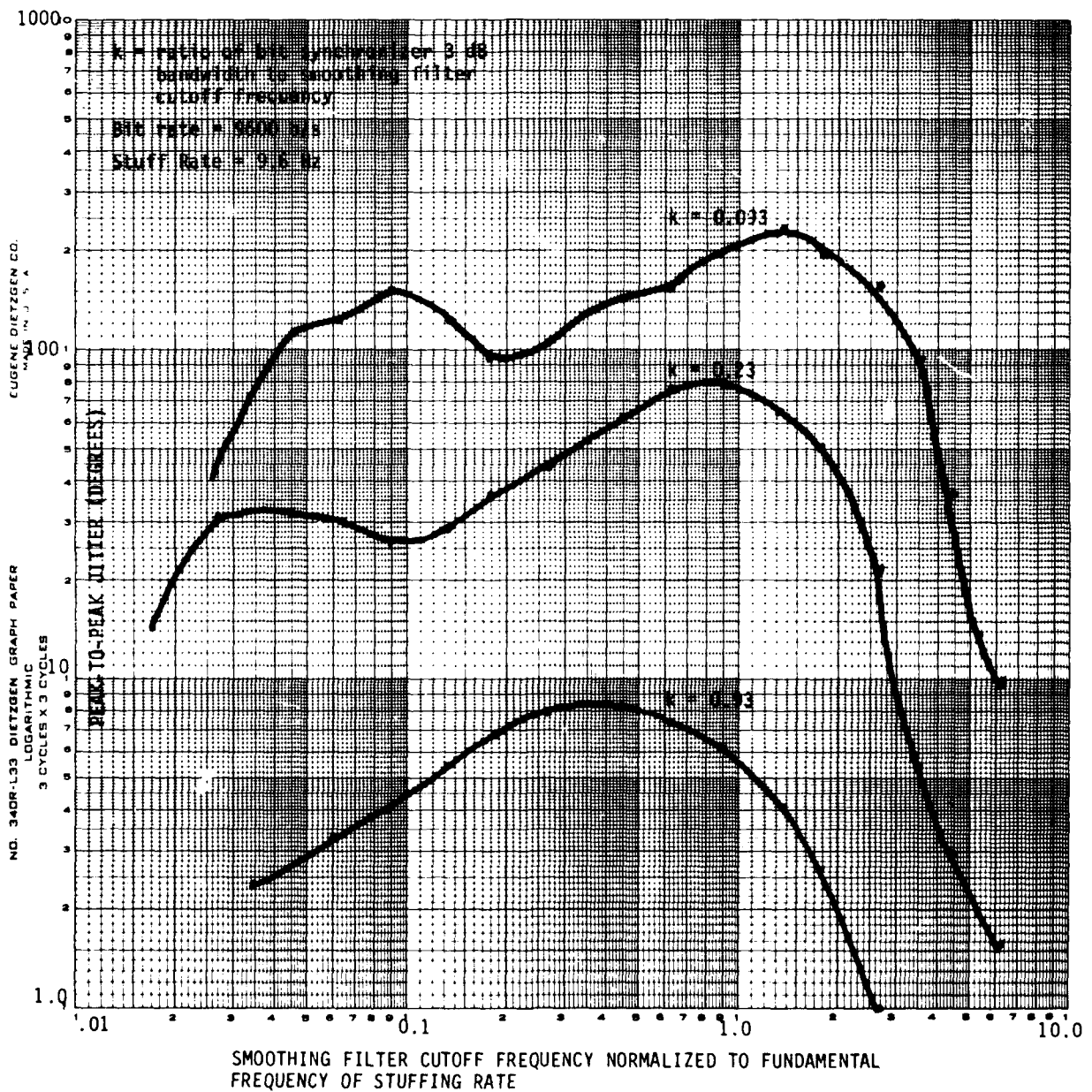
Figure 23. Effect of Pulse Stuffing Jitter on Clock Recovery for a Second Order Bit Synchronizer

SMOOTHING FILTER CUTOFF FREQUENCY NORMALIZED TO FUNDAMENTAL FREQUENCY OF STUFFING RATE

46

SMOOTHING FILTER CUTOFF FREQUENCY NORMALIZED TO
FUNDAMENTAL FREQUENCY OF STUFFING RATE
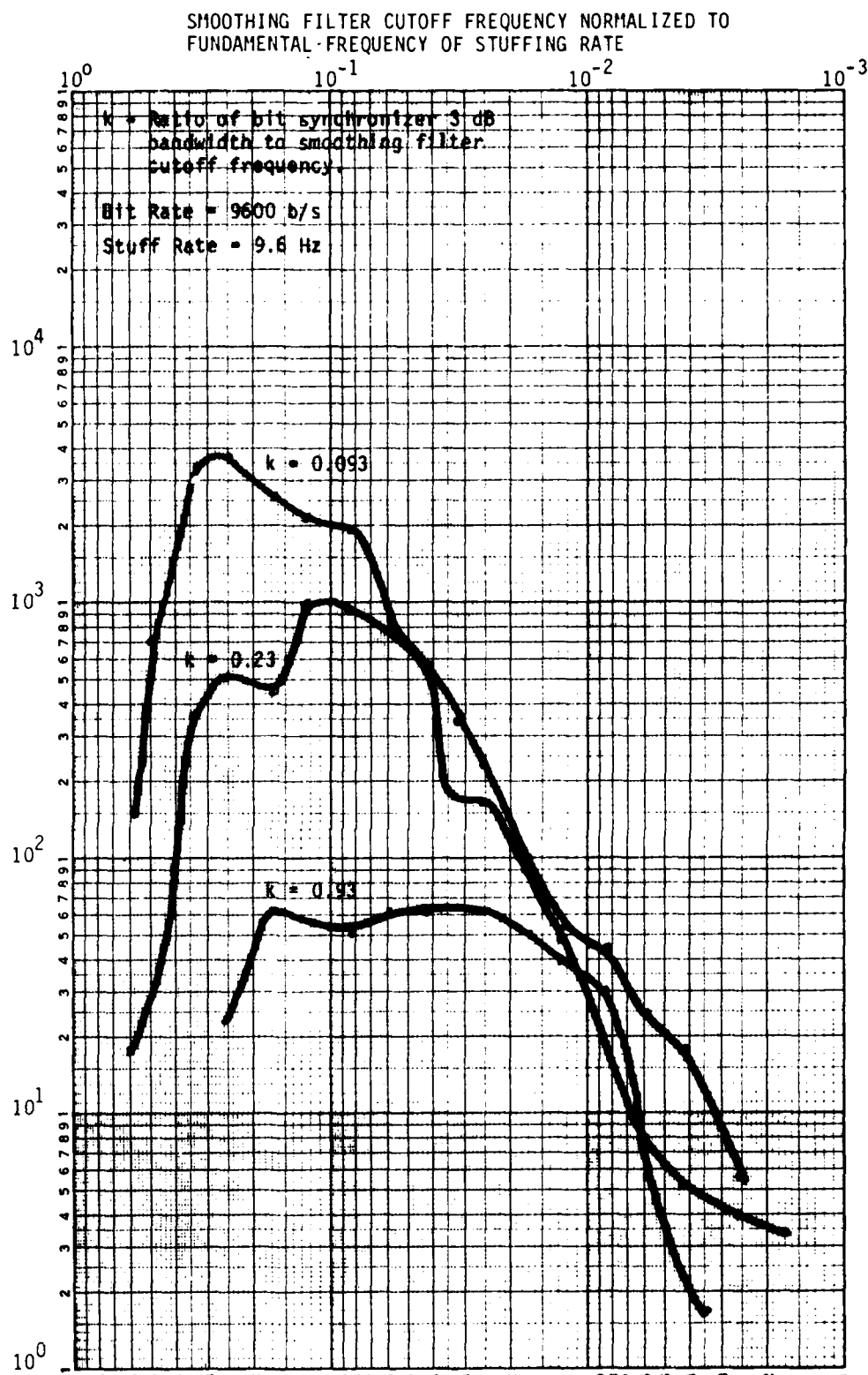
Figure 24. Effect on Pulse Stuffing Slew Rate on Clock Recovery
for a Second Order Bit Synchronizer

47

REFERENCES

1. Final Report, DCA Contract DCA-100-76-C-0072, "Analysis of Jitter Characteristics in a Pulse Stuffing TDM Network", June 1977.

2. D. R. Smith and J. L. Osterholz, "The Effects of Digital Tropo Error Statistics on Asynchronous Digital System Design", Record of the 1975 NTC, pp. (28-25) - (28-31).

3. D. L. Duttweiler, "Waiting Time Jitter", BSTJ Vol. 51, January 1972

4. Floyd M. Gardner, Phaselock Techniques, John Wiley and Sons, Inc. New York, 1967.

5. DCEC Ltr, R430, "Request for Test and Performance Evaluation of Commercial Narrowband Modem's Digital Interface Capability with the DSCS Equipment AN/GSC-24", 17 May 1979.

6. CSC Report, "History of Slew Specifications", 2 October 1979.

7. RADC Ltr, DCLD, "Compatibility Testing Between Production Model AN/GSC-24(V) and HN-74", 15 July 1975

8. T.O. 31W2-2GSC-24-2, AN/GSC-24 Theory of Operation, Installation and Maintenance.

9. DCEC Final Report, R200, "Pulse Code Modulation, Time Division Multiplex System Design Verification Test Program", February 1972.

10. A. Alan and B. Pritsker, The GASP IV Simulation Language, John Wiley and Sons, Inc., 1974.

APPENDIX A

USER'S GUIDE

1. INTRODUCTION

User familiarity with the ITEL AS-5 Batch and Time Sharing Option (TSO) Operating Systems including the WYLBUR Text Editor is assumed.

2. AVAILABILITY

These programs which include the Stuff/Fill bit and GASP Simulations are available for use on the DCEC ITEL AS-5 computer system.

3. STUFF/FILL BIT PROGRAM DATA CARD FORMAT

The following data cards are required to use STUFFILL:

Card #1:   Channel Information Card

Variable	Column position	Format
NCHANS	not specified	free
NCHAN	not specified	free

Card #2:   Stuff/fill Card

Variable	Column positions	Format
NFLBT	1 to 70 by 5	14I5

Card #3   Fill bits/channel Card

Variable	Column positions	Format
NFLBTC	1 to 70 by 5	14I5

Card #4   Stuffs/channel Card

Variable	Column positions	Format
NSTUFF	1 to 70 by 5	14I5

Card #5   Wait time jitter Card

Variable	Column positions	Format
NWTJ	1 to 70 by 5	14I5

Card #6   Port and word Card

Variable	Column positions	Format
NPORTS	not specified	free
NWORDS	not specified	free

Card #7   Port to channel Card

Variable	Column positions	Format
IPTCHN	1 to 70 by 5	14I5

4. STUFFIL USAGE

Each run, corresponding to the generation of the stuff/fill bit locations in the GSC-24's combined bit stream, requires all seven input cards described above.

To run STUFFIL, enter WYLBUR and execute the PDS member R3893.LIB(NIHGO). A typical session is listed in Appendix B. The input data for this program typically is saved in a partitioned data set (PDS) whose member name is specified by the user. All Fortran source programs are listed in Appendix D.

SAMPLE DIALOG TO USE STUFF/FILL PROGRAM

This dialog is valid only after the user has successfully logged
onto TSO and has received the READY response from the computer. For
this example, the user name is Cormack, the charge number is
11352000, and the user's badge number is R3893. The stuff/fill bit
locations in the GSC24's combined data stream will be determined as
a function of the inputs described in the User's Guide. User
responses follow the question marks.

```
W
COMMAND? execute from 'r3893.lib(mihgo)' clear
VOLUME IS TSOWK1
3 CHAR JOB IDENT?  sfb
RUN TIME(SEC)? 60
DATA DSN MEMBER NAME? gsc24a
GSC24.F8XX.DATA NUMBER? 7
DATA DSN MEMBER NAME?
//R3893SFB JOB (11352000,2G05,60,8,192),'JOHN J CORMACK',
// NOTIFY=R3893,MSGCLASS=Q
//PRINT1A EXEC PGM=UPRINT,REGION=60K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD  DSN=R3893.DATA(GSC24A),DISP=SHR
//PROGRUN EXEC PGM=GSC24,REGION=192K
//STEPLIB DD DSN=R3893.LOAD,DISP=SHR
//FT04F001 DD SYSOUT=A
//FT05F001 DD DSN=R3893.DATA(GSC24A),DISP=SHR,LABEL=(,,,IN)
//FT06F001 DD SYSOUT=A
//FT08F001 DD DSN=R3893.GSC24.F807.DATA,DISP=SHR
//FT09F001 DD DSN=&TEMP1,DISP=(NEW,DELETE),SPACE=(CYL,1),
// DCB=(RECFM=VS),UNIT=SYSDA
//FT10F001 DD DSN=&TEMP2,DISP=(NEW,DELETE),SPACE=(CYL,1),
// DCB=(RECFM=VS),UNIT=SYSDA
//FT11F001 DD DSN=&TEMP3,DISP=(NEW,DELETE),SPACE=(CYL,1),
// DCB=(RECFM=VS),UNIT=SYSDA
//FT12F001 DD DSN=&TEMP4,DISP=(NEW,DELETE),SPACE=(CYL,1),
// DCB=(RECFM=VS),UNIT=SYSDA
//PRINT1B EXEC PGM=UPRINT,REGION=60K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=R3893.GSC24.F807.DATA,DISP=SHR
//
RUN JOB(Y OR N)? y
JOB SUBMITTED
EXEC END
```

APPENDIX C

SAMPLE DIALOG TO USE THE GASP PROGRAM

This dialog is valid only after the user has successfully logged
onto TSO and has received the READY response from the computer. For
this example the user's name is Cormack, the charge number is
11352000, and the user's badge number is B3893. The first WYLBUR
exec file sets up the input data as required by the GASP
subroutines. The second WYLBUR exec file executes the program in
the background.

A. FIRST WYLBUR EXEC FILE

```
w
COMMAND? execute from 'r3893.lib(gaspm0d)' clear
VOLUME IS TSOWK1
DSN? data
MEMBER NAME? gasp96ab
J J CORMACK        24    12    27 1979     1000000000000011
J J CORMACK        24    06    10 1980      100000000000011

            2SSYNC        1     1        0.        520.
LOWER VALUE? -360.
UPPER VALUE? 360.
            2SSYNC        1     1     -360.        360.

            300UTPUT      1     1        0.        360.
LOWER VALUE? -180.
UPPER VALUE? 180.
            300UTPUT      1     1     -180.        180.

     4     0    1.0E-7    1.0E-8    1.0E-8    1.0E-4      -1.0
ABSOLUTE LOCAL TRUNCATION ERROR?
RELATIVE ERROR?
MIN STEP SIZE?
MAX STEP SIZE?
     4     0    1.0E-7    1.0E-8    1.0E-8    1.0E-4      -1.0

            1        7813        5
FRAME START NUMBER? 1
FRAME STOP NUMBER? 500
            1         500        5

9.58907E-3            1
NBPO? 25
9.58907E-3           25

            4           2
NODE TYPE? 2
            4           2
9.58907E-3           25
```

```
            9.6E-3              09.58907E-3
         9.6001E-3              09.58907E-3
         9.5999E-3              09.58907E-3
            9.6E-3              09.58907E-3
            9.5E-3
      T1 RATE? 9.6e-3
            9.6E-3          25
         9.6001E-3           0      9.6E-3
         9.5999E-3           0      9.6E-3
            9.6E-3           0      9.6E-3
            9.6E-3           0      9.6E-3
            9.6E-3

            9.6E-3           0      9.6E-3
      T2 RATE? 9.6001e-3
         9.6001E-3           0      9.6E-3

         9.6001E-3           0      9.6E-3
      T2 RATE? 9.6002e-3
         9.6002E-3           0      9.6E-3

         9.5999E-3           0      9.6E-3
      T2 RATE? 9.6004e-3
         9.6004E-3           0      9.6E-3

            9.6E-3           0      9.6E-3
      T2 RATE? 9.6006e-3
         9.6006E-3           0      9.6E-3

            0.0             90.0
            0.0             90.0
            0.0             90.0
            0.0             90.0
      INITIAL PHASE OF OUTPUT? 45.0
            0.0             45.0
            0.0             45.0
            0.0             45.0
            0.0             45.0

         1.8293E-3      .230701
         1.8293E-3      .230701
         1.8293E-3      .230701
         1.8293E-3      .230701
      FN? 1.8294e-3
         1.8294E-3
         1.8294E-3
         1.8294E-3
         1.8294E-3
      ZETA? .230701
         1.8294E-3      .230701
         1.8294E-3      .230701
         1.8294E-3      .230701
```

```
   1.8294E-3   .230701
PREFIX - DATA : MEMBER - GASP96AB
NONE = NO SAVE, STOP = PROGRAM END
MEMBER NAME? gzsp96ba
MEMBER GZSP96BA SAVED IN R3893.DATA
DSN? stop
EXEC END


B. SECOND WYLBUR EXEC FILE


w
COMMAND? execute from 'r3893.lib(gaspgo)' clear
VOLUME IS TSOWK1
3 CHAR IDENTIFIER? run
RUN TIME (SEC)? 120
OUTPUT LINES(K)? 5
PDS OR SEQ? pds
DATA INPUT MEMBER NAME? gzsp96ba
GSC24.F8XX.DATA FILE NUMBER? 7
LIST INPUT DATA SETS? y
# OF PLOTS? 1
# OF CYL FOR TEMP DSN? 3
BIT RATE (INTEGER)? 9600
PLOT EVERY XTH POINT? 25
POS OR NEG STUFF? pos
MORE RUNS? y
//R3893RUN JOB (11352000,2G05,120,5,150),'JOHN J CORMACK',
// MSGCLASS=Q,NOTIFY=R3893
//PRINTAO EXEC PGM=UPRINT,REGION=60K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=R3893.DATA(GZSP96BA),DISP=SHR
//PRINTBO EXEC PGM=UPRINT,REGION=60K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=R3893.GSC24.F807.DATA,DISP=SHR
//GO96001 EXEC PGM=GASPSIM,REGION=150K,TIME=(2,0)
//STEPLIB DD DSN=R3893.GASP.LOAD,DISP=SHR
//FT03F001 DD *
25,-1
/*
//FT04F001 DD DSN=R3893.GSC24.F807.DATA,DISP=SHR
//FT05F001 DD DSN=R3893.DATA(GZSP96BA),DISP=SHR,
// LABEL=(,,,IN)
//FT06F001 DD SYSOUT=A
//FT11F001 DD DSN=&GPLOT1,DISP=(NEW,DELETE),SPACE=(CYL,3),
// DCB=(RECPM=VS),UNIT=SYSDA
//
RUN JOB(Y OR N)? y
JOB SUBMITTED
EXEC END
```

APPENDIX D

STUFF/FILL SOURCE LISTING

```
C     DETERMINES WHICH DATA WORD CONTAINS FILL BITS AND WHERE IN THE
C     DATA WORD THE FILL BITS ARE LOCATED
      DIMENSION IFALAG(31),IWORD(32),MULT(10),NBFT(10),NBIT(10),NBYT(10)
      DIMENSION IPTCHN(31),NFLBTC(31),MBFT(5),MBIT(5),MBYT(5),NFLBT(31)
      DIMENSION NPTPCH(31),NSTUFF(31),NZERO(14),NDELPH(14),NCLOCK(14)
      INTEGER NWTJ(31),AREV,AAREV
      DATA MULT/1,2,4,8,16,32,64,128,256,512/
      DATA NWTJ/31*0/,NSTUFF/31*0/,NPTPCH/31*0/
C     NCHANS IS THE TOTAL NUMBER OF CHANNELS
C     NCHAN IS THE CHANNEL UNDER INVESTIGATION
      READ(5,*,END=999) NCHANS,NCHAN
C     NFLBTC IS THE ARRAY THAT TELLS HOW MANY FILL BITS ARE IN EACH
C     CHANNEL VIA A SPECIAL CODE WORD FOR EACH CHANNEL
      READ(5,120) (NFLBT(IAB),IAB=1,NCHANS)
      READ(5,120) (NFLBTC(IAB),IAB=1,NCHANS)
      READ(5,120) (NSTUFF(IAB),IAB=1,NCHANS)
      READ(5,120) (NWTJ(IAB),IAB=1,NCHANS)
120   FORMAT(14I5)
C     NPORTS IS THE TOTAL NUMBER OF PORTS
      READ(5,*) NPORTS,NWORDS
C     WRITE(13,510)
510   FORMAT(' FORTRAN FILE 13')
C     WRITE(14,610)
610   FORMAT(' FORTRAN FILE 14')
      WRITE(6,190) NPORTS,NCHANS,NWORDS,NSTUFF(NCHAN),NWTJ(NCHAN),NCHAN
190   FORMAT(I3,' PORTS, ',I2,' CHANNELS, ',I3,' WORDS AND ',I5,
     1' CLOCKS/STUFF AND ',I5,' CLOCKS/WAIT TIME FOR CHAN',I3)
      IF(NPORTS.LT.15.OR.NPORTS.GT.31) GO TO 999
      ICLOCK=0
      LTOTAL=0
      ITOTAL=0
      KFLAG=0
      MMWORD=0
      NPTS=0
      READ(5,120) (IPTCHN(IAD),IAD=1,NPORTS)
      DO 130 IAC=1,NCHANS
      WRITE(6,140) IAC,NFLBTC(IAC),NFLBT(IAC)
140   FORMAT(1X,'CHANNEL ',I2,' WHOSE FILL BIT CODE IS ',I4,' HAS ',I3,
     1' FILL BITS')
130   CONTINUE
      DO 150 IAC=1,NPORTS
      WRITE(6,160) IAC,IPTCHN(IAC)
160   FORMAT(1X,'PORT ',I2,' IS ASSOCIATED WITH CHANNEL ',I2)
150   CONTINUE
      DO 205 IAC=1,NCHANS
      DO 205 IACC=1,NPORTS
      IF(IPTCHN(IACC).EQ.IAC) NPTPCH(IAC) = NPTPCH(IAC) + 1
205   CONTINUE
```

```
            DO 210 IAC=1,NCHANS
            WRITE(6,215) IAC,NPTPCH(IAC)
215         FORMAT(' CHANNEL',I3,' HAS ',I2,' PORTS')
210         CONTINUE
            LCLOCK=NPTPCH(NCHAN)
C     NN IS THE DATA WORD POINTER
            MKNT=0
            DO 535 NN=1,NWORDS
            N=NN
            AN=FLOAT(N)
            DO 555 KNX=1,10
            NBFT(KNX)=0
            NBIT(KNX)=0
555         NBYT(KNX)=0
            I=0
10          I=I+1
            NDIV=N/2
            NBIT(I)=N-2*NDIV
            N=NDIV
            IF(NDIV.EQ.0) GOTO 20
            GOTO 10
C     CALCULATES THE BINARY VALUE OF A IE THE WHOLE NUMBER PART
20          DO 30 JJJJ=1,10
30          NBYT(JJJJ)=NBIT(11-JJJJ)
            ANFRCT=(AN-AINT(AN))
            DO 40 KKKKK=1,10
            A=ANFRCT*2.
            IF(A.GE.1.0) GOTO 35
            NBFT(KKKKK)=0
            ANFRCT=A
            GOTO 40
C     CALCULATES THE BINARY VALUE OF A IE THE DECIMAL PART
35          NBFT(KKKKK)=1
            ANFRCT=A-1.
40          CONTINUE
            AREV=0
C     CALCULATES THE REVERSE BINARY VALUE OF A
            DO 67 NNDEX=1,10
67          AREV=AREV+NBYT(NNDEX)*MULT(NNDEX)
            DO 151 IAA=1,32
            IF(IAA.GT.31) GO TO 145
            IFALAG(IAA)=0
145         IWORD(IAA)=0
151         CONTINUE
C     IAE IS THE CHANNEL POINTER
            IAE=NCHAN
C     AT EACH DATA WORD CHECK FOR FILL BIT INSERTIONS
            WRITE(6,401) NN,AREV,IAE,NFLBTC(IAE),NBIT,NBYT
 401        FORMAT(' NN=',I3,2X,'AREV=',I4,2X,'CHANNEL ',I2,' FILL BIT CODE
           1',I4,' BIN=',10I1,' REV BIN=',10I1)
            IF(AREV.GT.NFLBTC(IAE)) IFALAG(IAE)=1
            NUMB1=NPTPCH(IAE)
```

```
         DO 235 LADD=1,NUMB1
         IF(LADD.GT.1) GO TO 240
         IF(IFALAG(IAE).EQ.1) IWORD(LADD)=1
         IF(IFALAG(IAE).EQ.1) MKNT=MKNT+1
         IF(IFALAG(IAE).EQ.0) IWORD(LADD)=0
         GO TO 235
240      IWORD(LADD)=0
235      CONTINUE
         IFALAG(IAE)=0
C        ************************************************************
C        A DATA WORD HAS BEEN GENERATED
C        ************************************************************
         NUMB2=NPTPCH(NCHAN)
         DO 110 KOA=1,NUMB2
         LTOTAL=LTOTAL+1
         IF(MOD(LTOTAL,NSTUFF(NCHAN)).NE.0) GO TO 110
         IWORD(KOA)=IWORD(KOA)+1
110      CONTINUE
         NUMB3=NPTPCH(NCHAN)
         WRITE(6,200) NN,(IWORD(IAF),IAF=1,NUMB3)
200      FORMAT(' DATA WORD ',I3,' IS ',32I1)
C        ICNT = 0 WHEN FIRST FILL BIT PER WORD
C        ICNT = 1 WHEN NOT FIRST FILL BIT PER WORD
         ICNT=0
C        DO LOOP SAMPLES THRU ALL BITS OF DATA WORD
         NUMB4=NPTPCH(NCHAN)
         DO 50 JJC=1,NUMB4
         IF(IWORD(JJC).LT.1) GO TO 180
C        ICLOCK IS POINTER TO ARRAY CONTAINING FILL BIT INTERVALS
         ICLOCK=ICLOCK+1
C        IF(ICNT.EQ.0) WRITE(6,405) ICNT,JJC,LCLOCK,MMWORD
405      FORMAT(' ICNT=',I2,5X,'JJC=',I4,5X,'LCLOCK=',I3,5X,'MMWORD=',I3)
         IF(ICNT.EQ.0) NCLOCK(ICLOCK)=JJC+(NPTPCH(NCHAN)-LCLCCK)+MMWOBD
         IF(ICNT.EQ.0) MMWORD=0
         IF(KFLAG.EQ.0.AND.ICNT.EQ.0) IOFSET=NCLOCK(ICLOCK)
C        IF(ICNT.EQ.1) WRITE(6,500) ICNT,JJC,LCLOCK
500      FORMAT(' ICNT=',I5,5X,'JJC=',I5,5X,'LCLOCK=',I5)
         IF(ICNT.EQ.1) NCLOCK(ICLOCK)=JJC-LCLOCK
         NDELPH(ICLOCK)=IWORD(JJC)
         IF(ICLOCK.NE.1) GO TO 2222
         ISTORE=NDELPH(ICLOCK)
C2222    WRITE(6,2333) ICLOCK,NDELPH(ICLOCK),ICLOCK,NCLOCK(ICLOCK),JJC,IWOR
C        1D(JJC)
 2333    FORMAT(' NDELPH(',I2,')=',I2,5X,'NCLOCK(',I2,')=',I2,5X,'IWOBD(',I
        12,')=',I2)
2222     KFLAG=1
         LCLOCK=JJC
         ICNT=1
230      IF(ICLOCK.LT.14) GO TO 55
220      WRITE(9) (NCLOCK(IAH),IAH=1,ICLOCK)
C        WRITE(13,505) (NCLOCK(IAH),IAH=1,ICLOCK)
505      FORMAT(14I5)
```

D-3

```fortran
      WRITE(10) (NDELPH(IAH),IAH=1,ICLOCK)
C     WRITE(14,606) (NDELPH(IAH),IAH=1,ICLOCK)
606   FORMAT(14I5)
80    FORMAT(14I5)
      DO 250 IAZ=1,ICLOCK
250   ITOTAL=ITOTAL+NCLOCK(IAZ)
      NPTS=NPTS+14
      ICLOCK=0
      GO TO 55
180   IF(JJC.EQ.NPTPCH(NCHAN).AND.ICNT.EQ.0) MMWORD=MMWORD+NPTPCH(NCHAN)
55    IF(NN.NE.NWORDS) GO TO 50
      IF(JJC.NE.NPTPCH(NCHAN)) GO TO 50
C     LAST WORD CASE
225   ICLOCK=ICLOCK+1
C     IF(ICNT.EQ.0) WRITE(6,410) ICNT,LCLOCK,MMWORD,IOFSET
410   FORMAT(' ICNT=',I3,5X,'LCLOCK=',I3,5X,'MMWORD=',I3,5X,'IOFSET=',I3
     1)
      IF(ICNT.EQ.0) NCLOCK(ICLOCK)=(NPTPCH(NCHAN)-LCLOCK)+MMWORD+5*NPTPC
     1H(NCHAN)+IOFSET
C     IF(ICNT.EQ.1) WRITE(6,415) ICNT,JJC,LCLOCK,IOFSET
415   FORMAT(' ICNT=',I3,5X,'JJC=',I3,5X,'LCLOCK=',I3,5X,'IOFSET=',I3)
      IF(ICNT.EQ.1) NCLOCK(ICLOCK)=JJC-LCLOCK+5*NPTPCH(NCHAN)+IOFSET
      NDELPH(ICLOCK)=IWORD(JJC)
      IF(NDELPH(ICLOCK).EQ.0) NDELPH(ICLOCK)=ISTORE
C     WRITE(6,2432) ICLOCK,NDELPH(ICLOCK),ICLOCK,NCLOCK(ICLOCK),JJC,IWOR
C     1D(JJC)
2432  FORMAT(' LAST WORD....NDELPH(',I2,')=',I2,5X,'NCLOCK(',I2,')=',I2,
     15X,'IWORD(',I2,')=',I2)
      WRITE(9) (NCLOCK(IAH),IAH=1,ICLOCK)
C     WRITE(13,505) (NCLOCK(IAH),IAH=1,ICLOCK)
      WRITE(10) (NDELPH(IAH),IAH=1,ICLOCK)
C     WRITE(14,606) (NDELPH(IAH),IAH=1,ICLOCK)
      DO 260 IAZ=1,ICLOCK
260   ITOTAL=ITOTAL+NCLOCK(IAZ)
      NPTS=NPTS+ICLOCK
      GO TO 535
50    CONTINUE
535   CONTINUE
      WRITE(6,305) MKNT
305   FORMAT(' TOTAL NUMBER OF FILL BITS IS ',I5)
      WRITE(6,245) NPTS
245   FORMAT(' NPTS=',I5)
      END FILE 9
      REWIND 9
      END FILE 10
      REWIND 10
      JCOUNT=0
      ICOUNT=0
      NTOT=0
      NPTSS=0
      IFLAG=0
      DO 1000 IA=1,NPTS,14
```

```
        NDIFF=NPTS-IA
        NDIFF1=NDIFF+1
        IF(NDIFF.GE.13) NDIFF1=14
        READ(9)  (NFLBT(IAD),IAD=1,NDIFF1)
        READ(10)  (IFALAG(IAD),IAD=1,NDIFF1)
        IF(IA.NE.1.AND.NDIFF.GE.13) NLASTP=NLAST
        IF(NDIFF.GE.13) NLAST=NFLBT(14)
        DO 1015 IB=1,NDIFF1
        NTOTP=NTOT
        IF(NFLBT(IB).NE.1) NTOT=0
        NTOT=NTOT+IFALAG(IB)
        IF(NFLBT(IB).EQ.1) IFLAG=1
        IF(NFLBT(IB).EQ.1) GO TO 1015
        ICOUNT=ICOUNT+1
        NCLOCK(ICOUNT)=NFLBT(IB)
        IF(NDIFF.LE.13.AND.IB.EQ.NDIFF1) GO TO 1045
        IF(ICOUNT.LT.14) GO TO 1025
1045    WRITE(11) (NCLOCK(IAD),IAD=1,ICOUNT)
        ICOUNT=0
1025    IF(IB.EQ.1) GO TO 1035
        IBM1=IB-1
        NPREV=NFLBT(IBM1)
        GOTO 1040
1035    IF(IA.NE.1) NPREV=NLASTP
        IF(IA.EQ.1) NPREV=1
1040    IF(IFLAG.EQ.0) GO TO 1030
        IF(NFLBT(IB).NE.1.AND.NPREV.EQ.1) GO TO 1030
        GOTO 1015
1030    IF(IFLAG.EQ.0) JCOUNT=JCOUNT+1
        IF(IFLAG.EQ.0) NDELPH(JCOUNT)=NTOT
        IF(IFLAG.EQ.1) NDELPH(JCOUNT)=NTOTP
        IF(IFLAG.EQ.1) JCOUNT=JCOUNT+1
        IF(IFLAG.EQ.1) NDELPH(JCOUNT)=NTOT
        IFLAG=0
        IF(NDIFF.LE.13.AND.IB.EQ.NDIFF1) GO TO 1050
        IF(JCOUNT.LT.14) GOTO 1015
1050    WRITE(12) (NDELPH(IAD),IAD=1,JCOUNT)
        NPTSS=NPTSS+JCOUNT
C       WRITE(6,82) NPTSS,JCOUNT,NDIFF1,IA,IB
  82    FORMAT(' NPTSS =',I5,5X,' JCOUNT =',I5,
     15X,' NDIFF1 =',I5,5X,'IA =',I3,5X,'IB =',I3)
        JCOUNT=0
1015    CONTINUE
1000    CONTINUE
        WRITE(6,85) NPTSS
85      FORMAT(' NPTSS =',I5)
        WRITE(8,81) NPTSS
81      FORMAT(I10)
        IF(NPTSS.EQ.0) WRITE(6,83)
83      FORMAT(' PROGRAM TERMINATED SINCE NPTSS=0 ')
        WRITE(6,270) ITOTAL
270     FORMAT(' THE TOTAL NUMBER OF CLOCK PERIODS IS',I7)
```

```
                    IF(NPTSS.EQ.0) GO TO 999
                    END FILE 11
                    REWIND 11
                    END FILE 12
                    REWIND 12
                    DO 300 IAH=1,NPTSS,14
                    NDIFF=NPTSS-IAH
                    NDIFF1=NDIFF+1
                    IF(NDIFF.GE.13) NDIFF1=14
                    READ(11) (NCLOCK(IAD),IAD=1,NDIFF1)
                    WRITE(8,80) (NCLOCK(IAD),IAD=1,NDIFF1)
     300            CONTINUE
                    DO 545 IWOL=1,14
     545            NZERO(IWOL)=NWTJ(NCHAN)
     C              WRITE(6,585) NZERO(14)
     585            FORMAT(' NZERO(14) =',I5)
     C              WRITE(6,80) (NZERO(IADDD),IADDD=1,14)
                    DO 340 IAH=1,NPTSS,14
                    NDIFF=NPTSS-IAH
                    NDIFF1=NDIFF+1
                    IF(NDIFF.GE.13) NDIFF1=14
     C              WRITE(6,84) NDIFF1
     84             FORMAT(' NDIFF1=',I5)
                    WRITE(8,80) (NZERO(IAD),IAD=1,NDIFF1)
     340            CONTINUE
                    DO 320 IAH=1,NPTSS,14
                    NDIFF=NPTSS-IAH
                    NDIFF1=NDIFF+1
                    IF(NDIFF.GE.13) NDIFF1=14
     C              WRITE(6,800) NDIFF1
      800           FORMAT(' AT READ FILE 12...NDIFF1 =',I5)
                    READ(12) (NDELPH(IAD),IAD=1,NDIFF1)
                    WRITE(8,80) (NDELPH(IAD),IAD=1,NDIFF1)
     320            CONTINUE
                    WRITE(4,81) NPTS
                    REWIND 9
                    REWIND 10
                    DO 400 IAH=1,NPTS,14
                    NDIFF=NPTS-IAH
                    NDIFF1=NDIFF+1
                    IF(NDIFF.GE.13) NDIFF1=14
                    READ(9) (NCLOCK(IAD),IAD=1,NDIFF1)
                    WRITE(4,80) (NCLOCK(IAD),IAD=1,NDIFF1)
     400            CONTINUE
                    DO 425 IAH=1,NPTS,14
                    NDIFF=NPTS-IAH
                    NDIFF1=NDIFF+1
                    IF(NDIFF.GE.13) NDIFF1=14
                    WRITE(4,80) (NZERO(IAD),IAD=1,NDIFF1)
     425            CONTINUE
                    DO 420 IAH=1,NPTS,14
                    NDIFF=NPTS-IAH
```

```
      NDIPP1=NDIFP+1
      IP(NDIPP.GE.13) NDIFF1=14
      READ(10) (NDELPH(IAD),IAD=1,NDIFF1)
      WRITE(4,80) (NDELPH(IAD),IAD=1,NDIFF1)
420   CONTINUE
999   STOP
      END
```

GASP USER SOURCE LISTING

```
C*****PULSE JITTER STUDY**************************************************
C
C      PURPOSE
C          SIMULATE CASCADED NODES FOR ANALYSIS OF WAITING TIME JITTER
C
C      DESCRIPTION OF PARAMETERS
C          DW2( ) - DATA TRANSFER FREQUENCY (RAD/SEC) AT EACH NODE.
C                      PERTURBATION FROM BASIC CHANNEL FREQUENCY.
C          DW3( ) - VCO FREQUENCY (RAD/SEC) AT EACH NODE.
C                      PERTURBATION FROM BASIC CHANNEL FREQUENCY.
C          GAIN( )- GAIN OF PHASELOCK LOOP.
C          TAU( ) - TIME CONSTANT FOR SECOND ORDER PHASELOCK LOOP.
C          PHERR( )-SAMPLED PHASE ERROR (RAD).
C          TS( )  - TIME BETWEEN SAMPLES FOR PHASELOCK LOOP. PERIOD OF
C                      SYNC FREQUENCY FOR EACH NODE.
C          NFD( ) - FIXED DELAY IN PHASELOCK LOOP CORRECTION.
C                      DELAY WITHIN FRAME IN BITS.
C          CHFREQ - CHANNEL FREQUENCY.
C          NBPO   - NUMBER OF BITS PER OPPORTUNITY.
C          NFCUR  - CURRENT FRAME BEING PROCESSED.
C          NFSTRT - STARTING FRAME FOR DATA COLLECTION.
C          NFSTP  - STOPPING FRAME FOR DATA COLLECTION.
C          DTF    - TIME INCREMENT USED TO DETERMINE FSLEW.
C          NSpF   - NUMBER OF SAMPLES PER FRAME.
C          IDECSN - FLAG FOR DECISION TIME. RESET BY SAVE EVENT.
C
C      SYSTEM EVENTS
C          SAVE   - SAVE TIME FOR DATA.
C                      EVENT OCCURS EVERY SAVE TIME DTSAV.
C                      EVENT CODE 1.
C          DCRCT  - OPPORTUNITY TO DECIDE TO CORRECT PHASE OF SYNC
C                      FREQUENCY FOR NODE. OCCURS EVERY NBPF BITS.
C                      EVENT CODE 2.
C          CRCTN  - TIME TO CORRECT PHASE OF SYNC FREQUENCY FOR NODE.
C                      THIS EVENT GENERATED FROM DCRCT EVENT.
C                      EVENT CODE 3.
C          SEVNT  - STATE EVENT. DETECTION OF OUTPUT PHASE MINIMUM
C                      OR MAXIMUM. EVENT CODE 4.
C          FSAVE  - SAVE FREQUENCY TO DETERMINE FSLEW.
C                      EVENT CODE 5.
C
C      FILES
C          FILE 1 - EVENT FILE.
C                      ATRIB(1) - TIME OF EVENT.
C                      ATRIB(2) - EVENT CODE.
C                      ATRIB(3) - NODE INVOLVED IN EVENT ACTIVITY.
C
C      DATA COLLECTION
```

```
C          COLCT.
C              1 - PHMAX.
C              2 - PHMIN.
C          TIMSA.
C              1 - PHOUT.
C              2 - FNODE.
C              3 - FSLEW.
C              4 - PHERRN.
C              5 - PHTRK.
C              6 - PHSRCE.
C              7 - FTRK.
C              8 - FSLEWT.
C          HISTOGRAMS
C              1 - PHMAX.
C              2 - PHMIN.
C              3 - PHOUT.
C              4 - PHERRN.
C              5 - PHTRK.
C              6 - PHSRCE.
C              7 - FNODE.
C              8 - FSLEW.
C          PLOTS
C           PLOT 1.
C              3 - PHOUT.
C              4 - PHSRCE.
C              5 - PHERRN.
C              6 - PHTRK.
C           PLOT 2.
C              1 - FNODE.
C              2 - FSLEW.
C
C     REMARKS
C         THIS MODEL IS A PERTURBATIONAL MODEL.
C         DATA IS COLLECTED AT LAST NODE IN CASCADE.
C
C     SUBPROGRAMS REQUIRED
C         INTLC,EVNTS,STATE,PHASE + GASP SUBPROGRAMS
C
C***************************************************************************
C
C     NAME MUXJIT
C
C     PROGRAM DATE 780417
C
      REAL*16 QF2
      COMMON QSET(200)
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,N
     1NAPO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TTBEG
     2,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /SYSTEM/ NODES,NTYPE,DW2(11),DW3(11),GAIN(11),TAU(11),
     1 PHERR(11),NFD(11),TS(11),CHFREQ,PHSYNP,PHSYNN,PHSYNC,NPLOMI,
     2 MNOP,TPI,NBPO,NFSTRT,NFSTP,NFCUR,DTSAVE,DTF,IDECSN,NSFF,INIT,
```

```
      3 NNCNT,QF2,NUMB(1000),NSEQ(1000),NFILBT,NCNT,F2,F3,NWTJ(1000)
C
       TPI = 6.283185
       NNCNT=0
       NCRDR=5
       NPRNT=6
C
       CALL GASP
       STOP
C
       END
       BLOCK DATA
       REAL*16 QF2
       COMMON /SYSTEM/ NODES,NTYPE,DW2(11),DW3(11),GAIN(11),TAU(11),
      1 PHERR(11),NPD(11),TS(11),CHFREQ,PHSYNP,PHSYNN,PHSYNC,NPLOMI,
      2 MNOP,TPI,NBPO,NFSTRT,NFSTP,NFCUR,DTSAVE,DTF,IDECSN,NSPF,INIT,
      3 NNCNT,QF2,NUMB(1000),NSEQ(1000),NFILBT,NCNT,F2,F3,NWTJ(1000)
       DATA NSEQ/1000*0/,NUMB/1000*0/,NWTJ/1000*0/
       END
C*****SUBROUTINE EVNTS*********************************************************
C
C      PURPOSE
C         PROCESS SIMULATION EVENTS
C
C      DESCRIPTION OF PARAMETERS
C         NSYNC  - POINTER TO NODE N SYNC PHASE STATE.
C         NPHSE  - POINTER TO NODE N OUTPUT PHASE STATE.
C         NSRCE  - POINTER TO NODE N SOURCE PHASE STATE.
C         NINT   - POINTER TO NODE N INTERMEDIATE STATE.
C         PHSRCE - PHASE OF NODE N SOURCE.
C         PHSYNC - PHASE OF NODE N SYNC.
C         PHOUT  - PHASE OF NODE N OUTPUT.
C         PHERRN - PHASE OF NODE N ERROR.
C         PHTRK  - PHASE OF TRACKING LOOP ERROR.
C         FNODE  - FREQUENCY OF NODE N VCO.
C         FSLEW  - RATE OF CHANGE OF FREQUENCY OF NODE N VCO.
C         FTRK   - FREQUENCY OF TRACKING NODE VCO.
C      FSLEWT    - RATE OF CHANGE OF FREQUENCY OF TRACKING NODE VCO.
C
C      SUBPROGRAMS REQUIRED
C         STATE,PHASE + GASP SUBPROGRAMS
C
C*****************************************************************************
C
       SUBROUTINE EVNTS(IX)
C
C      SUBPROGRAM DATE 780417
C
       DIMENSION XX(10)
       REAL*16 CHFREQ,AA1,AA2,QF2
       COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,N
      1NAPO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TTBEG
```

```
2,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW,ISEES,LFLAG(50),NFLAG,
     1NNEQD,NNEQS,NNEQT,SS(100),SSL(100),TTNEX
      COMMON /GCOM3/ AAERR,DTMAX,DTMIN,DTSAV,IITES,LLERR,LLSAV,LLSEV,RRE
     1RR,TTLAS,TTSAV
      COMMON /GCOM4/ DTPLT(10),HHLOW(25),HHWID(25),IICRD,IITAP(10),JJCEL
     1(500),LLABC(25,2),LLABH(25,2),LLABP(11,2),LLABT(25,2),LLPHI(10),LL
     2PLO(10),LLPLT,LLSUP(15),LLSYM(10),MMPTS,NNCEL(25),NNCLT,NNHIS,NNPL
     3T,NNPTS(10),NNSTA,NNVAR(10),PPHI(10),PPLO(10)
      COMMON /GCOM6/ EENQ(100),IINN(100),KKRNK(100),MMAXQ(100),QQTIM(100
     1),SSOBV(25,5),SSTPV(25,6),VVNQ(100)
      COMMON /SYSTEM/ NODES,NTYPE,DW2(11),DW3(11),GAIN(11),TAU(11),
     1 PHERR(11),NFD(11),TS(11),CHFREQ,PHSYNP,PHSYNN,PHSYNC,NPLOMI,
     2 MNOP,TPI,NBPO,NFSTRT,NFSTP,NFCUR,DTSAVE,DTF,IDECSN,MSPF,INIT,
     3 NNCNT,QF2,NUMB(1000),NSEQ(1000),NPILBT,NCNT,F2,F3,NWTJ(1000)
C
C     *****
C     *
C******          DETERMINE NODE FOR EVENT ACTIVITY AND ESTABLISH
C     *          CORRECT POINTERS FOR PHASE
C     *
C     *****
C
      DATA NNCNT/0/,NNFLAG/0/
      N = ATRIB(3)
      IF (IX .EQ. 4) N = NODES
      CALL STATE
      CALL PHASE(NTYPE,N,NSYNC,NPHSE,NINT,PHSYNC,PHOUT,PHINT,PHSRCE)
      PHSYNP=PHSYNN
      PHSYNN=PHSYNC
      IF(NCNT.EQ.1) NPSEQ=0
      IF(NCNT.GT.1) NCNTM1=NCNT-1
      IF(NCNT.GT.1) NPSEQ=NSEQ(NCNTM1)
      IF(NNFLAG.EQ.1.AND.IX.NE.2) GO TO 4012
      IF(NNFLAG.EQ.1) GO TO 3001
      IF(NFCUR.LT.3.OR.ABS(PHSYNN).GE.ABS(PHSYNP)) GO TO 4012
3001  NNFLAG=0
      IF(NSEQ(NCNT).EQ.0) NNFLAG=1
      IF(NSEQ(NCNT).EQ.0) GO TO 5568
      AA1=(QF2*1.0Q3)/QFLOAT(NBPO*NSEQ(NCNT))
      IF(NWTJ(NCNT).EQ.0) GO TO 310
      AA2=1.056250Q0/QFLOAT(NWTJ(NCNT))
      GO TO 320
310   AA2=0.Q0
320   QHFREQ=(QF2*1.0Q3+QFLOAT(NPLOMI)*AA1*(1.0Q0+AA2))/1.0Q3
      CHFREQ=SNGLQ(QHFREQ)
      F3=CHFREQ
      GO TO 5569
5568  CHFREQ=F2
5569  F3=CHFREQ
      NNODES=NODES + 1
C     WRITE(NPRNT,5045) TNOW,CHFREQ,F2
```

```fortran
5045  PORMAT(' TNOW=',E14.7,5X,'CHPREQ=',1PE14.7,5X,'P2=',E14.7)
C     IF(TNOW.LE.5.E-4) WRITE(NPRNT,4054) NCNT,CHFREQ,NSEQ(NCNT),
C    1NNODES
4054  PORMAT(' IN EVNTS....IX=4....NCNT=',I3,2X,'CHFREQ=',E20.7,2X,
     1'NSEQ()=',I5,/2X,' NNODES=',I3)
      DO 14110 IJKNUM=1,NNODES
      IF(IJKNUM.LE.NODES) DW2(IJKNUM)=1.E6*(P2-CHFREQ)*TPI
14110 DW3(IJKNUM)=1.E6*(P3-CHFREQ)*TPI
C
C     *****
C     *
C******          PROCESS EVENT CODE IX
C     *
C     *****
C
C4012 IF(TNOW.LE.5.E-4) WRITE(NPRNT,4050) CHFREQ,TNOW
4050  PORMAT(/' IN EVNTS......CHFREQ = ',E20.6,2X,' TNOW = ',E16.5)
4012  GO TO (1000,2000,3000,4000,5000), IX
C
C     *****
C     *
C******          SAVE
C     *
C     *****
C
 1000 CONTINUE
C     IF(TNOW.LE.5.E-4) WRITE(NPRNT,4055) IDECSN,PHSYNC,PHOUT,PHSRCE,
C    1PHERRN,PHTRK
4055  PORMAT(' IN EVNTS..IX=1',2X,'IDECSN=',I2,2X,'PHSYNC=',E13.5,
     12X,'PHOUT=',E13.5,/,' PHSRCE=',E13.5,2X,'PHERRN ',E13.5,2X,
     2'PHTRK=',E13.5)
C
C******          SAVE DATA POR PLOTTING AND STATISTICS COLLECTION
C     *          AT REGULAR INTERVALS
C
      TSAVE = TNOW
      PHSRCE = PHSRCE * 360.0 / TPI
      PHSYNC = PHSYNC * 360.0 / TPI
      PHOUT = PHOUT * 360.0 / TPI
      PHERRN = PHSYNC - PHOUT
      PHTRK = PHOUT - SS(NNEQD-1) * 360.0 / TPI
      PNODE = DD(NPHSE) / TPI
      PTRK = DD(NNEQD-1) / TPI
C
      IF (INIT .EQ. 0) GO TO 1010
C
C******          FIRST SAVE TIME
C     *          CLEAR STATISTICAL ARRAYS AND INITIALIZE
C
      CALL CLEAR
      SSTPV(1,6) = PHOUT
      SSTPV(4,6) = PHERRN
```

```
         SSTPV(5,6)  =  PHTRK
         SSTPV(6,6)  =  PHSRCE
C
 1010 CONTINUE
C
      IF (NNPLT .EQ. 0) GO TO 1100
C
C******              STORE DATA FOR PLOTTING
C
      XX( 1)  =  IDECSN
      XX (2)  =  PHSYNC
      XX ( 3)  =  PHOUT
      XX (4)  =  PHSRCE
      XX ( 5)  =  PHERRN
      XX (6)  =  PHTRK
      IDECSN  =  0
      NNCNT= NNCNT+1
      IF(NNOP.EQ.1)  CALL GPLOT(XX,TNOW,1)
      IF (MOD(NNCNT,MNOP).EQ.1) CALL GPLOT(XX,TNOW,1)
C
 1100 CONTINUE
      IF (NNSTA .GT. 0) CALL TIMSA(PHOUT,TNOW, 1)
      IF (NNSTA .GT. 3) CALL TIMSA(PHERRN,TNOW,4)
      IF (NNSTA .GT. 4) CALL TIMSA(PHTRK,TNOW,5)
      IF (NNSTA .GT. 5) CALL TIMSA(PHSRCE,TNOW,6)
      IF (NNHIS .GT. 2) CALL HISTO(PHOUT,3)
      IF (NNHIS .GT. 3) CALL HISTO(PHERRN,4)
      IF (NNHIS .GT. 4) CALL HISTO(PHTRK,5)
      IF (NNHIS .GT. 5) CALL HISTO(PHSRCE,6)
C
C******              SCHEDULE NEXT FREQUENCY SAVE TIME TO GET FSLEW
C
      ATRIB(1)  =  TNOW + DTF
      ATRIB(2)  =  5.0
      CALL FILEM(1)
      RETURN
C
C      *****
C         *
C******              DCRCT
C         *
C      *****
C
 2000 CONTINUE
      IF (N .EQ. NODES) NFCUR = NFCUR + 1
C     IF(TNOW.LE.5.E-4) WRITE(NPRNT,4056)N,NODES,NFCUR
4056  FORMAT(' IN EVNTS...IX=2...N=',I3,2X,'NODES=',I3,2X,'NFCUR=',
     1 I3)
      IF (NFCUR .GT. NFSTP) MSTOP = -1
C
C******              SCHEDULE NEXT DECISION FOR CORRECTION
C
```

E-6

```
            ATRIB(1) = TNOW + NBPO * TS(N)
            CALL FILEM(1)
C
            IDECSN = 1
            IF ((NFCUR .LT. NFSTRT) .OR. (N .NE. NODES)) GO TO 2500
C
C******             CREATE NSPF SAVE TIMES FOR THIS FRAME
C
            ATRIB(2) = 1.0
            ATRIB(3) = N
            ATRIB(4) = 0.0
            DO 2100 I = 1,NSPF
            ATRIB(1) = TNOW + (I - 1) * DTSAVE + NFD(N) * TS(N)
            CALL FILEM(1)
 2100 CONTINUE
C
C******             TEST FOR 1 BIT ERROR BETWEEN SOURCE CHANNEL AND
C      *            FREQUENCY SOURCE PHASE
C
 2500 CONTINUE
            ERR = PHSYNC - PHSRCE
            IF(ABS(ERR).LT.TPI) RETURN
C
C******             SCHEDULE PHASE CORRECTION
C      *            ATTRIBUTE 4 IS THE CORRECTION TO BE MADE
C      *            CORRECT PHASE IMMEDIATELY IF NFD(N) = 0
C
            ATRIB(4) = SIGN(TPI,ERR)
            IF (NFD(N) .EQ. 0) GO TO 3000
            ATRIB(1) = TNOW + NFD(N) * TS(N)
            ATRIB(2) = 3
            CALL FILEM(1)
            RETURN
C
C      *****
C      *
C******             CRCTN
C      *
C      *****
C
 3000 CONTINUE
C     IF (TNOW.LE.5.E-4) WRITE(NPRNT,4057)
 4057 FORMAT(' IN EVNTS...IX=3')
C
C******             CORRECT PHASE OF SYNC FREQUENCY FOR NODE N
C
C     IF(TNOW.LE.5.E-4) WRITE(NPRNT,4999) NCNT,NUMB(NCNT)
 4999 FORMAT(' NUMB(',I3,')=',I5)
            ANUMBR=NUMB(NCNT)
            SS(NSYNC) = SS(NSYNC) - ANUMBR * ATRIB(4)
            NCNT=NCNT+1
            IF(NCNT.GT.NFILBT) NCNT=1
```

```
      RETURN
C
C       *****
C       *
C******             STATE EVENT
C       *
C       *****
C
 4000 CONTINUE
C     IF (TNOW.LE.5.E-4) WRITE(NPRNT,4051)
 4051 FORMAT(' IN EVNTS....IX=4')
      PHOUT = PHOUT * 360.0 / TPI
      IF (LFLAG(1) .EQ. 0) GO TO 4010
C     PHASE MAXIMUM
      IF (NNHIS .GT. 0) CALL HISTO(PHOUT,1)
      IF (NNCLT .GT. 0) CALL COLCT(PHOUT,1)
C     IF (TNOW.LE.5.E-4) WRITE(NPRNT,4052)
 4052 FORMAT(' IN EVNTS....IX=4...PHASE MAX')
      GO TO 4020
C
 4010 CONTINUE
      IF (LFLAG(2) .EQ. 0) GO TO 4020
C     PHASE MINIMUM
      IF (NNHIS .GT. 1) CALL HISTO(PHOUT,2)
      IF (NNCLT .GT. 1) CALL COLCT(PHOUT,2)
C     IF (TNOW.LE.5.E-4) WRITE(NPRNT,4053)
 4053 FORMAT(' IN EVNTS....IX=4...PHASE MIN')
C
 4020 CONTINUE
      RETURN
C
C       *****
C       *
C******             FREQUENCY SAVE EVENT
C       *
C       *****
C
 5000 CONTINUE
C     IF (TNOW.LE.5.E-4) WRITE(NPRNT,4058)
 4058 FORMAT(' IN EVNTS...IX=5')
      PSLEW = (DD(NPHSE) / TPI - FNODE) / (TNOW - TSAVE)
      PSLEWT = (DD(NNEQD-1) / TPI - FTRK) / (TNOW - TSAVE)
      IF (INIT .EQ. 0) GO TO 5010
      INIT = 0
      SSTPV(2,6) = FNODE
      SSTPV(3,6) = PSLEW
      SSTPV(7,6) = FTRK
      SSTPV(8,6) = PSLEWT
C
 5010 CONTINUE
      IF (NNPLT .EQ. 1) GO TO 5100
C
```

```
C******              STORE DATA FOR PLOTTING
C
      XX(1) = FNODE
      XX(2) = FSLEW
      XX(3) = FTRK
      XX(4) = FSLEWT
      MMCMT=MMCMT+1
      IF (MNOP.EQ.1) CALL GPLOT(XX,TSAVE,2)
      IF(MOD(MMCMT,MNOP).EQ.1) CALL GPLOT(XX,TSAVE,2)
C
 5100 CONTINUE
      IF (NNSTA .GT. 1) CALL TIMSA(FNODE,TSAVE,2)
      IF (NNSTA .GT. 2) CALL TIMSA(FSLEW,TSAVE,3)
      IF (NNSTA .GT. 6) CALL TIMSA(FTRK,TSAVE,7)
      IF (NNSTA .GT. 7) CALL TIMSA(FSLEWT,TSAVE,8)
      IF (NNHIS .GT. 6) CALL HISTO(FNODE,7)
      IF (NNHIS .GT. 7) CALL HISTO(FSLEW,8)
      RETURN
C
      END
C*****SUBROUTINE INTLC********************************************************
C
C     PURPOSE
C        INITIALIZE SYSTEM
C
C     REMARKS
C        CODE IS INCLUDED TO INITIALIZE SYSTEM IN STEADY STATE.
C
C     SUBPROGRAMS REQUIRED
C        PHASE + GASP SUBPROGRAMS
C
C***************************************************************************
C
      SUBROUTINE INTLC
C
C     SUBROUTINE DATE 770221
      DIMENSION PSYNC(11),POUT(11)
      REAL*16 QF2,AA1,AA2,QHFREQ
C
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,N
     1NAPO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NERNT,PPARM(50,4),TNOW,TTBEG
     2,TTCLR,TTFIN,TTRIB(25),ITSET
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW,ISEES,LFLAG(50),NFLAG,
     1NNEQD,NNEQS,NNEQT,SS(100),SSL(100),TTNEX
      COMMON /GCOM3/ AAERR,DTMAX,DTMIN,DTSAV,IITES,LLERR,ILSAV,LLSEV,RRE
     1RR,TTLAS,TTSAV
      COMMON /GCOM4/ DTPLT(10),HHLOW(25),HHWID(25),IICRD,IITAP(10),JJCEL
     1 (500),LLABC(25,2),LLABH(25,2),LLABP(11,2),LLABT(25,2),LLPHI(10),LL
     2PLO(10),LLPLT,LLSUP(15),LLSYM(10),MMPTS,NNCEL(25),NNCLT,NNHIS,NNPL
     3T,NNPTS(10),NNSTA,NNVAR(10),PPHI(10),PPLC(10)
      COMMON /SYSTEM/ NODES,NTYPE,DW2(11),DW3(11),GAIN(11),TAU(11),
     1 PHERR(11),NFD(11),TS(11),CHFREQ,PHSYNP,PHSYNN,PHSYNC,NPLOMI,
```

```
      2 MNOP,TPI,NBPO,NFSTRT,NFSTP,NFCUR,DTSAVE,DTF,IDECSN,NSPF,INIT,
      3 NNCNT,QF2,NUMB(1000),NSEQ(1000),NFILBT,NCNT,F2,F3,NWTJ(1000)
C
        WRITE(NPRNT,9700)
 9700   FORMAT('1')
C
        NCNT=1
        PHSYNN=0.
        NNEQD = 0
        NFCUR = 1
        MCRDR=NCRDR-1
        LCRDR=NCRDR-2
        READ(NCRDR,101) NFSTRT,NFSTP,NSPF
 101    FORMAT(3I10)
        READ(LCRDR,*) MNOP,NPLOMI
        WRITE(NPRNT,9100) NFSTRT,NFSTP,NSPF
 9100   FORMAT(10X,'DATA COLLECTION STARTS AT FRAME',I4,
      1 1X,'AND STOPS AT FRAME ',I5,1X,
      2 'WITH ',I4,' SAMPLES PER FRAME ')
C
C       *****
C       *
C******              READ INPUT PARAMETERS AND CONVERT TO SIMULATION
C       *            INPUT FORMS
C       *
C       *****
C
C******              ENTER CHANNEL FREQUENCY IN MHZ AND NUMBER
C       *            OF BITS PER OPPORTUNITY
C
        READ(NCRDR,102) CHFREQ,NBPO
 102    FORMAT(F10.0,I10)
C
C******              ENTER NUMBER OF NODES AND TYPE OF SYSTEM
C
        READ(NCRDR,103) NODES,NTYPE
 103    FORMAT(2I10)
        WRITE(NPRNT,9120) NODES,NTYPE
 9120   FORMAT(///,10X,'SYSTEM PARAMETERS FOR',I3,1X,'NODES'/
      1 10X,'ALL NODES ARE TYPE',I2,1X,'SYSTEMS')
        READ(MCRDR,110) NFILBT
 110    FORMAT(I10)
        READ(MCRDR,120) (NSEQ(IAD),IAD=1,NFILBT)
        READ(MCRDR,120) (NWTJ(IAD),IAD=1,NFILBT)
        READ(MCRDR,120) (NUMB(IAD),IAD=1,NFILBT)
 120    FORMAT(14I5)
        WRITE(6,140)
 140    FORMAT(/10X,'NSEQ ARRAY LISTED')
        WRITE(NPRNT,125) (NSEQ(JAL),JAL=1,NFILBT)
 125    FORMAT(10X,14I5)
        WRITE(6,145)
 145    FORMAT(/10X,'NWTJ ARRAY LISTED')
```

```
      WRITE(NPRNT,125) (NWTJ(JAL),JAL=1,NFILBT)
      WRITE(6,130)
130   FORMAT (/10X,'NUMB ARRAY LISTED')
      WRITE(NPRNT,125) (NUMB(JAL),JAL=1,NFILBT)
C
C******              ENTER PARAMETER SETS FOR EACH NODE
C
      DO 1000 N = 1,NODES
      WRITE(NPRNT,9130) N
 9130 FORMAT (//,10X,'NODE',I3)
C
C******              ENTER DATA TRANSFER FREQUENCY IN MHZ, FIXED DELAY
C     *              IN BITS AND VCO NOMINAL FREQUENCY IN MHZ
C
      READ(NCRDR,104) QF2,NFD(N),F3
104   FORMAT(Q10.3,I10,F10.0)
      F2=SNGLQ(QF2)
      IF(N.NE.1) GO TO 112
      AA1=(QF2*1.0Q3)/QFLOAT(NBPO*NSEQ(1))
      IF(NWTJ(1).EQ.0) GO TO 310
      AA2=1.05625Q0/QFLOAT(NWTJ(1))
      GO TO 320
310   AA2=0.0Q0
320   QHFREQ=(QF2*1.0Q3+QFLOAT(NPLOMI)*AA1*(1.0Q0+AA2))/1.0Q3
      CHFREQ=SNGLQ(QHFREQ)
112   F3=CHFREQ
      IF(N.NE.1) GO TO 111
      WRITE(NPRNT,9110) CHFREQ,NBPO,MNOP
 9110 FORMAT(////,10X,'T1 FREQUENCY SOURCE =',1PE15.7,1X,'MHZ'/
     1 10X,'NUMBER OF BITS PER OPPORTUNITY =',I4/10X,
     2 'EVERY',I6,'TH DATA POINT IS PLOTTED')
111   WRITE(NPRNT,9140) F2,NFD(N),F3
 9140 FORMAT (/,10X,'T2 FREQUENCY SOURCE =',1PE15.7,1X,'MHZ'/
     1 10X,'CORRECTION APPLIED IN BIT',I3,1X,'OF FRAME'/
     1 10X,'VCO NOMINAL FREQUENCY =',1PE15.7,1X,'MHZ')
      TS(N) = 1.0 / (F2 * 1.0E6)
      DW2(N) = 1.0E6 * (F2 - CHFREQ) * TPI
      DW3(N) = 1.0E6 * (F3 - CHFREQ) * TPI
C
C******              ENTER SYNC PHASE AND OUTPUT PHASE IN DEG
C
      READ(NCRDR,105) PSYNC(N),POUT(N)
105   FORMAT(2F10.0)
      WRITE(NPRNT,9145) PSYNC(N),POUT(N)
 9145 FORMAT (/,10X,'T2 INITIAL PHASE =',F5.0,1X,'DEG'/
     1 10X,'NODE OUTPUT PHASE =',F5.0,1X,'DEG')
      PSYNC(N) = PSYNC(N) * TPI / 360.0
      POUT(N) = POUT(N) * TPI / 360.0
C
C******              LOOP BANDWIDTH IN HZ AND DAMPING (SECOND ORDER ONLY)
C
      IF (NTYPE .EQ. 2) GO TO 100
```

```
C
C      FIRST ORDER LOOP
C
       READ(NCRDR,106) FN
106    FORMAT(F10.0)
       WRITE(NPRNT,9150) FN
 9150  FORMAT(/,10X,'NATURAL FREQUENCY (FN) =',1PE15.7,1X,'HZ')
       WN = TPI * FN
C      CONVERT BANDWIDTH INPUT TO GAIN FOR SIMULATION
       GAIN(N) = WN
       NNEQD = NNEQD + 2
       GO TO 200
C
C      SECOND ORDER LOOP
C
  100 CONTINUE
       READ(NCRDR,105) FN,ZETA
       WRITE(NPRNT,9160) FN,ZETA
 9160  FORMAT(/,10X,'NATURAL FREQUENCY (FN) =',1PE15.7,1X,'HZ'/
      1 10X,'DAMPING (ZETA) =',0PF5.3)
       WN = TPI * FN
C      CONVERT BANDWIDTH INPUT AND DAMPING TO GAIN AND TAU FOR SIMULATION
       GAIN(N) = WN * WN
       TAU(N)  = 2 * ZETA / WN
       NNEQD = NNEQD + 3
C
  200 CONTINUE
C
 1000 CONTINUE
C
C      *****
C      *
C******              TRACKING LOOP PARAMETERS
C      *
C      *****
C
C
C******              ENTER VCO NOMINAL FREQUENCY IN MHZ
C
       READ(NCRDR,106) F3
       F3=CHFREQ
       WRITE(NPRNT,9170) F3
 9170  FORMAT(///,10X,'TRACKING LOOP PARAMETERS'/
      1 10X,'VCO NOMINAL FREQUENCY =',1PE15.7,1X,'MHZ')
       DW3(N) = 1.0E6 * (F3 - CHFREQ) * TPI
C
C******              ENTER LOOP BANDWIDTH IN HZ AND DAMPING
C
       READ(NCRDR,105) FN,ZETA
       WRITE(NPRNT,9160) FN,ZETA
       WN = TPI * FN
C      CONVERT BANDWIDTH INPUT AND DAMPING TO GAIN AND TAU FOR SIMULATION
```

E-12

```
          GAIN(N) = WN * WN
          TAU(N) = 2 * ZETA / WN
          NNEQD = NNEQD + 2
          NNEQT = NNEQD
C
C     *****
C       *
C******              INITIALIZE SYSTEM EVENTS
C       *
C     *****
C
          DO 2000 N = 1,NODES
          CALL PHASE(NTYPE,N,NSYNC,NPHSE,NINT,PHSYNC,PHOUT,PHINT,PHSRCE)
          IF (NTYPE .EQ. 2) GO TO 1100
C
C     FIRST ORDER SYSTEM
C
C     INITIALIZE PHASES
          SS(NSYNC) = PHSRCE + PSYNC(N)
          SS(NPHSE) = SS(NSYNC) + POUT(N)
          GO TO 1200
C
C     SECOND ORDER SYSTEM
C
 1100 CONTINUE
C     INITIALIZE PHASES
          SS(NSYNC) = PHSRCE + PSYNC(N)
          SS(NPHSE) = SS(NSYNC) + POUT(N)
          SS(NINT) = -DW3(N) / GAIN(N)
 1200 CONTINUE
          PHERR(N) = 0.0
C
C******              SCHEDULE FIRST DCRCTN TIME
C
          ATRIB(1) = 0.0
          ATRIB(2) = 2.0
          ATRIB(3) = N
          ATRIB(4) = 0.0
          CALL FILEM(1)
 2000 CONTINUE
C
          N = NODES + 1
          NINPUT = NPHSE
          NPHSE = NNEQD - 1
          NINT = NNEQD
          SS(NPHSE) = SS(NINPUT)
          SS(NINT) = -DW3(N) / GAIN(N)
          IDECSN = 0
          INIT = 1
          DTSAVE = FLOAT(NBPO) * TS(NODES) / FLOAT(NSPF)
          DTPLT(1) = DTSAVE*FLOAT(MNOP)
          DTPLT(2) = DTSAVE*FLOAT(MNOP)
```

```
        DTP = 0.1 * TS(NODES)
        RETURN
        END
C*****SUBROUTINE PHASE*********************************************************
C
C     PURPOSE
C        DETERMINE POINTERS AND CURRENT VALUES FOR NODE SYNC AND
C        OUTPUT PHASE BASED ON TYPE OF NODE AND NODE NUMBER
C
C     SUBPROGRAMS REQUIRED
C        NONE
C
C******************************************************************************
C
        SUBROUTINE PHASE(NTYPE,NODE,NSYNC,NPESE,NINT,PHSYNC,PHOUT,PHINT,
     1 PHSRCE)
C
C     SUBROUTINE DATE 761214
C
        COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW,ISEES,LFLAG(50),NFLAG,
     1 NNEQD,NNEQS,NNEQT,SS(100),SSL(100),TTNEX
C
        PHSRCE = 0.0
C
        GO TO (100,200), NTYPE
C
C     FIRST ORDER SYSTEM
C
  100 CONTINUE
        NSYNC = 2 * NODE - 1
        NPHSE = NSYNC + 1
        NSRCE = NSYNC - 1
        PHSYNC = SS(NSYNC)
        PHOUT = SS(NPHSE)
        IF (NSRCE .GT. 0) PHSRCE = SS(NSRCE)
        RETURN
C
C     SECOND ORDER SYSTEM
C
  200 CONTINUE
        NSYNC = 3 * NODE - 2
        NPHSE = NSYNC + 1
        NSRCE = NSYNC - 1
        NINT = NPHSE + 1
        PHSYNC = SS(NSYNC)
        PHOUT = SS(NPHSE)
        PHINT = SS(NINT)
        IF (NSRCE .GT. 0) PHSRCE = SS(NSRCE)
        RETURN
C
        END
C*****SUBROUTINE SCOND*********************************************************
```

```
C
C      PURPOSE
C          DETECT MINIMUM AND MAXIMUM PHASE PEAKS AT LAST NODE IN CASCADE
C
C      REMARKS
C          PHASE MIN AND MAX DETECTED WITHIN TWO DEGREES.
C
C      SUBPROGRAMS REQUIRED
C          GASP SUBPROGRAMS
C*************************************************************************
C
       SUBROUTINE SCOND
C
C      SUBROUTINE DATE 770110
C
       REAL*16 QP2
       COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,N
      1NAPO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TTBEG
      2,TTCLR,TTFIN,TTRIB(25),TTSET
       COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW,ISEES,LFLAG(50),NFLAG,
      1NNEQD,NNEQS,NNEQT,SS(100),SSL(100),TTNEX
       COMMON /SYSTEM/ NODES,NTYPE,DW2(11),DW3(11),GAIN(11),TAU(11),
      1 PHERR(11),NPD(11),TS(11),CHFREQ,PHSYNP,PHSYNN,PHSYNC,NPLOMI,
      2 MNOP,TPI,NBPO,NFSTRT,NFSTP,NFCUR,DTSAVE,DTF,IDECSN,NSPF,INIT,
      3 NNCNT,QP2,NUMB(1000),NSEQ(1000),NFILBT,NCNT,F2,F3,NWTJ(1000)
C
       GO TO (100,200), NTYPE
C
  100 CONTINUE
C     FIRST ORDER SYSTEM
       NPHSE = 2 * NODES
       GO TO 300
  200 CONTINUE
C     SECOND ORDER SYSTEM
       NPHSE = 3 * NODES - 1
  300 CONTINUE
       LFLAG(1) = KROSS(-NPHSE,0,0,0,-1,100.0)
       LFLAG(2) = KROSS(-NPHSE,0,0,0,1,100.0)
C
       RETURN
       END
C*****SUBROUTINE STATE***************************************************
C
C      PURPOSE
C          CALCULATE SYSTEM STATE DERIVATIVES
C
C      REMARKS
C          THE SYSTEM IS A CASCADE OF NODES FOLLOWED BY A TRACKING LOOP.
C          ALL NODES IN CASCADE ARE THE SAME TYPE (FIRST OR SECOND ORDER).
C          THE TRACKING LOOP IS A SECOND ORDER LCOP.
C
C      SUBPROGRAMS REQUIRED
```

E-15

```fortran
C        NONE
C
C***********************************************************************
C
      SUBROUTINE STATE
C
C      SUBROUTINE DATE 770221
C
      REAL*16 QF2
      COMMON /GCOM1/ ATRIB(25),JEVNT,MFA,MFE(100),MLE(100),MSTOP,NCRDR,N
     1NAPO,NNAPT,NNATR,NNFIL,NNQ(100),NNTRY,NPRNT,PPARM(50,4),TNOW,TTBEG
     2,TTCLR,TTFIN,TTRIB(25),TTSET
      COMMON /GCOM2/ DD(100),DDL(100),DTFUL,DTNOW,ISEES,LFLAG(50),NFLAG,
     1NNEQD,NNEQS,NNEQT,SS(100),SSL(100),TTNEX
      COMMON /SYSTEM/ NODES,NTYPE,DW2(11),DW3(11),GAIN(11),TAU(11),
     1 PHERR(11),NFD(11),TS(11),CHFREQ,PHSYNP,PHSYNN,PHSYNC,NPLOMI,
     2 MNOP,TPI,NBPO,NPSTRT,NPSTP,NPCUR,DTSAVE,DTF,IDECSN,NSPF,INIT,
     3 NNCNT,QF2,NUMB(1000),NSEQ(1000),NFILBT,NCNT,F2,F3,NWTJ(1000)
C
C      *****
C      *
C******          SOLVE FIRST OR SECOND ORDER SYSTEM EQUATIONS
C      *          FOR ALL NODES
C      *
C      *****
C
      IF (NTYPE .EQ. 2) GO TO 1000
C
C******          FIRST ORDER SYSTEM EQUATIONS
C
      DO 100 N = 1,NODES
      NSYNC = 2*N - 1
      NPHSE = NSYNC + 1
      PHERR(N) = SS(NSYNC) - SS(NPHSE)
      DD(NSYNC) = DW2(N)
      DD(NPHSE) = GAIN(N) * PHERR(N) + DW3(N)
  100 CONTINUE
C
      GO TO 2000
C
C******          SECCND ORDER SYSTEM EQUATIONS
C
 1000 CONTINUE
      DO 1100 N = 1,NODES
      NSYNC = 3 * N - 2
      NPHSE = NSYNC + 1
      NINT = NPHSE + 1
      PHERR(N) = SS(NSYNC) - SS(NPHSE)
      DD(NSYNC) = DW2(N)
      DD(NPHSE) = GAIN(N) * (TAU(N) * PHERR(N) + SS(NINT)) + DW3(N)
      DD(NINT) = PHERR(N)
 1100 CONTINUE
```

E-16

```
C
C          *****
C          *
C******              TRACKING LOOP
C          *
C          *****
C
 2000 CONTINUE
C
C******              SET UP POINTERS FOR TRACKING LOOP
C
      N = NODES + 1
      NINPUT = NPHSE
      NPHSE = NNEQD - 1
      NINT = NNEQD
      PHERR(N) = SS(NINPUT) - SS(NPHSE)
      DD(NPHSE) = GAIN(N) * (TAU(N) * PHERR(N) + SS(NINT)) + DW3(N)
      DD(NINT) = PHERR(N)
      RETURN
      END
```

APPENDIX F

WYLBUR PROGRAM SOURCE LISTING

A. DATA SET MEMBER 'R3893.LIB(NIHGO)' LISTING

```
1.  ; NIHGO
2.  SET EXE NOL TER
3.  CLR ACT
4.  SET * LAST
5.  SET ESC X
6.  @ N1 = 0
7.  READ STRING SO PROMPT'3 CHAR JOB IDENT? '
8.  READ VALUE NO PROMPT'RUN TIME(SEC)? '
9.  *+1 //R3893%SO JOB (1135200G,2G05,%NO,8,192),'JOHN J CORMACK',
10. *+1 // NOTIFY=R3893,MSGCLASS=Q
11. READ STRING S1 PROMPT'DATA DSN MEMBER NAME? '
12. IF('%S1' EQ '') X 39.000
13. @ N1 = N1 + 1
14. *+1 //PRINT%N1A EXEC PGM=UPRINT,REGION=60K
15. *+1 //SYSPRINT DD SYSOUT=A
16. *+1 //SYSUT1 DD DSN=R3893.DATA(%S1),DISP=SHR
17. *+1 //PROGRUN EXEC PGM=GSC24,REGION=192K
18. *+1 //STEPLIB DD DSN=R3893.LOAD,DISP=SHR
19. *+1 //FT04F001 DD SYSOUT=A
20. *+1 //FT05F001 DD DSN=R3893.DATA(%S1),DISP=SHR,LABEL=(,,,IN)
21. READ STRING S2 PROMPT'GSC24.F8XX.DATA NUMBER? '
22. IF('%S2' NE '') @ S2=LPAD(S2,2,'0')
23. *+1 //FT06F001 DD SYSOUT=A
24. IF('%S2' NE '') *+1 //FT08F001 DD DSN=R3893.GSC24.F8%S2.DATA,DISP=SH
25. IF('%S2' EQ '') *+1 //FT08F001 DD SYSOUT=A
26. *+1 //FT09F001 DD DSN=&TEMP1,DISP=(NEW,DELETE),SPACE=(CYL,1),
27. *+1 // DCB=(RECFM=VS),UNIT=SYSDA
28. *+1 //FT10F001 DD DSN=&TEMP2,DISP=(NEW,DELETE),SPACE=(CYL,1),
29. *+1 // DCB=(RECFM=VS),UNIT=SYSDA
30. *+1 //FT11F001 DD DSN=&TEMP3,DISP=(NEW,DELETE),SPACE=(CYL,1),
31. *+1 // DCB=(RECFM=VS),UNIT=SYSDA
32. *+1 //FT12F001 DD DSN=&TEMP4,DISP=(NEW,DELETE),SPACE=(CYL,1),
33. *+1 // DCB=(RECFM=VS),UNIT=SYSDA
34. IF('%S2' EQ '') X 11.000
35. *+1 //PRINT%N1B EXEC PGM=UPRINT,REGION=60K
36. *+1 //SYSPRINT DD SYSOUT=A
37. *+1 //SYSUT1 DD DSN=R3893.GSC24.F8%S2.DATA,DISP=SHR
38. X 11.000
39. *+1 //
40. LIST
41. READ STRING SO PROMPT'RUN JOB(Y OR N)? '
42. IF('%SO' NE 'Y') X 46.000
43. FRA
44. SUB UNN
45. JO
46. SET ESC
```

B. DATA SET MEMBER 'R3893.LIB(GASPMOD)' LISTING

```
1. ; GASPMOD
2. SET EXE NOL TER
3. SET ESC &
4. @ S0 = DATE
5. READ STRING S3 PROMPT'DSN? '
6. IF('&S3' EQ 'STOP') X 172.000
7. IF('&S3' EQ '') X 9.000
8. @ S6 = S3
9. READ STRING S4 PROMPT'MEMBER NAME? '
10. IF('&S4' EQ '') X 62.000
11. USE &S6(&S4) CLR
12. @ S1 = SUBSTR(S0,10,2)
13. @ S2 = SUBSTR(S0,13,2)
14. @ S5 = SUBSTR(S0,16,2)
15. LIST FIRST UNN
16. CH 24/25 TO '&S1' NOL
17. CH 29/30 TO '&S2' NOL
18. CH 34/35 TO '&S5' NOL
19. LIST * UNN
20. COMM
21. LIST 15 UNN
22. READ STRING S0 PROMPT'LOWER VALUE? '
23. IF('&S0' EQ '') X 32.000
24. @ S0 = LPAD(S0,10)
25. CH 31/40 TO '&S0' NOL
26. READ STRING S0 PROMPT'UPPER VALUE? '
27. IF('&S0' EQ '') X 30.000
28. @ S0 = LPAD(S0,10)
29. CH 41/50 TO '&S0' NOL
30. LIST * UNN
31. COMM
32. LIST 16 UNN
33. READ STRING S0 PROMPT'LOWER VALUE? '
34. IF('&S0' EQ '') X 43.000
35. @ S0 = LPAD(S0,10)
36. CH 31/40 TO '&S0' NOL
37. READ STRING S0 PROMPT'UPPER VALUE? '
38. IF('&S0' EQ '') X 41.000
39. @ S0 = LPAD(S0,10)
40. CH 41/50 TO '&S0' NOL
41. LIST * UNN
42. COMM
43. LIST 22 UNN
44. READ STRING S0 PROMPT'ABSOLUTE LOCAL TRUNCATION ERROR? '
45. IF('&S0' EQ '') X 48.000
46. @ S0 = LPAD(S0,10)
47. CH 11/20 TO '&S0' NOL
48. READ STRING S0 PROMPT'RELATIVE ERROR? '
49. IF('&S0' EQ '') X 52.000
```

```
50.  a S0 = LPAD(S0,10)
51.  CH 21/30 TO '&S0' NOL
52.  READ STRING S0 PROMPT'MIN STEP SIZE? '
53.  IF('&S0' EQ '') X 56.000
54.  a S0 = LPAD(S0,10)
55.  CH 31/40 TO '&S0' NOL
56.  READ STRING S0 PROMPT'MAX STEP SIZE? '
57.  IF('&S0' EQ '') X 60.000
58.  a S0 = LPAD(S0,10)
59.  CH 41/50 TO '&S0' NOL
60.  LIST * UNN
61.  COMM
62.  LIST 25 UNN
63.  READ STRING S0 PROMPT'FRAME START NUMBER? '
64.  IF('&S0' EQ '') X 73.000
65.  READ STRING S1 PROMPT'FRAME STOP NUMBER? '
66.  IF('&S1' EQ '') X 69.000
67.  a S1 = LPAD(S1,10)
68.  CH 11/20 TO '&S1' NOL
69.  a S0 = LPAD(S0,10)
70.  CH 1/10 TO '&S0' NOL
71.  LIST * UNN
72.  COMM
73.  LIST 26 UNN
74.  READ STRING S0 PROMPT'NBPO? '
75.  IF('&S0' EQ '') X 79.000
76.  a S0 = LPAD(S0,10)
77.  CH 11/20 TO '&S0' UNN
78.  COMM
79.  LIST 27 UNN
80.  READ STRING S0 USING 27 COL 1/10
81.  a S0 = LTRIM(S0)
82.  a N0 = S0
83.  READ STRING S0 PROMPT'NODE TYPE? '
84.  IF('&S0' EQ '') X 83.000
85.  a S7 '0'
86.  IF('&S0' EQ '2') a S7 '1'
87.  a S0 = LPAD(S0,10)
88.  CH 11/20 TO '&S0' UNN
89.  LIST 26 UNN
90.  IF(N0 EQ 1) a S9 '28'
91.  IF(N0 EQ 2) a S9 '28,31'
92.  IF(N0 EQ 3) a S9 '28,31,34'
93.  IF(N0 EQ 4) a S9 '28,31,34,37'
94.  IF(N0 EQ 5) a S9 '28,31,34,37,40'
95.  IF(N0 EQ 6) a S9 '28,31,34,37,40,43'
96.  IF(N0 EQ 7) a S9 '28,31,34,37,40,43,46'
97.  IF(N0 EQ 8) a S9 '28,31,34,37,40,43,46,49'
98.  LIST &S9 UNN
99.  a N0 = 28 + (N0 * 3)
100. LIST &N0 UNN
101. READ STRING S0 PROMPT'T1 RATE? '
```

```
102.  IF('&S0' EQ '')  X 110.000
103.  a S0 = LPAD(S0,10)
104.  CH 1/10 TO '&S0' IN 26 NOL
105.  CH 21/30 TO '&S0' IN  &S9 NOL
106.  CH 1/10 TO '&S0' IN &W0 NOL
107.  LIST 26 UNN
108.  LIST &S9 UNN
109.  LIST &W0 UNN
110.  COMM
111.  IF(N0 EQ 1) a W1 = 28
112.  IF(N0 EQ 2) a W1 = 31
113.  IF(N0 EQ 3) a W1 = 34
114.  IF(N0 EQ 4) a W1 = 37
115.  IF(N0 EQ 5) a W1 = 40
116.  IF(N0 EQ 6) a W1 = 43
117.  IF(N0 EQ 7) a W1 = 46
118.  IF(N0 EQ 8) a W1 = 49
119.  a W2 = 28
120.  LIST &W2 UNN
121.  READ STRING S0 PROMPT'T2 RATE? '
122.  IF('&S0' EQ '')  X 125.000
123.  a S0 = LPAD(S0,10)
124.  CH 1/10 TO '&S0' UNN IN &W2
125.  IF(W2 GE W1)  X 129.000
126.  a W2 = W2 + 3.
127.  COMM
128.  X 120.000
129.  COMM
130.  IF(N0 EQ 1) a S8 '29'
131.  IF(N0 EQ 2) a S8 '29,32'
132.  IF(N0 EQ 3) a S8 '29,32,35'
133.  IF(N0 EQ 4) a S8 '29,32,35,38'
134.  IF(N0 EQ 5) a S8 '29,32,35,38,41'
135.  IF(N0 EQ 6) a S8 '29,32,35,38,41,44'
136.  IF(N0 EQ 7) a S8 '29,32,35,38,41,44,47'
137.  IF(N0 EQ 8) a S8 '29,32,35,38,41,44,47,50'
138.  LIST &S8 UNN
139.  READ STRING S0 PROMPT'INITIAL PHASE OF OUTPUT? '
140.  IF('&S0' EQ '')  X 143.000
141.  a S0 = LPAD(S0,10)
142.  CH 11/20 TO '&S0' UNN IN &S8
143.  COMM
144.  IF(N0 EQ 1) a S5 '30'
145.  IF(N0 EQ 2) a S5 '30,33'
146.  IF(N0 EQ 3) a S5 '30,33,36'
147.  IF(N0 EQ 4) a S5 '30,33,36,39'
148.  IF(N0 EQ 5) a S5 '30,33,36,39,42'
149.  IF(N0 EQ 6) a S5 '30,33,36,39,42,45'
150.  IF(N0 EQ 7) a S5 '30,33,36,39,42,45,48'
151.  IF(N0 EQ 8) a S5 '30,33,36,39,42,45,48,51'
152.  LIST &S5 UNN
153.  READ STRING S0 PROMPT'FN? '
```

```
154. IF('&S0' EQ '') X 158.000
155. @ S0 = LPAD(S0,10)
156. CH 1/10 TO '&S0' IN &S5 NOL
157. LIST &S5 COL 1/10 UNN
158. IF('&S7' NE '1') X 163.000
159. READ STRING S0 PROMPT'ZETA? '
160. IF('&S0' EQ '') X 163.000
161. @ S0 = LPAD(S0,10)
162. CH 11/20 TO '&S0' IN &S5 NOL
163. LIST &S5 UNN
164. SH NAME
165. COMM NONE = NO SAVE, STOP = PROGRAM END
166. READ STRING S0 PROMPT'MEMBER NAME? '
167. IF('&S0' EQ 'NONE') X 4.000
168. IF('&S0' EQ 'STOP') X 172.000
169. IF('&S0' EQ '') S * CAR REP
170. IF('&S0' NE '') S *(&S0) CAR REP
171. X 4.000
172. SET ESC


   C. DATA SET MEMBER 'R3893.LIB(GASPGO)' LISTING

1. ; GASPGO
2. SET EXE NOL TER
3. SET ESC %
4. READ STRING S2 PROMPT'3 CHAR IDENTIFIER? '
5. IF('%S2' EQ '') X 71.000
6. CLR ACT
7. SET * LAST
8. @ W0 = LAST + 1
9. READ VALUE N1 PROMPT'RUN TIME (SEC)? '
10. @ N5 = 150
11. READ VALUE N7 PROMPT'OUTPUT LINES(K)? '
12. *+1 //R3893%S2 JOB (1135200G,2G05,%N1,%N7,%N5),'JOHN J CORMACK',
13. IF(N1 LE 600) *+1 // MSGCLASS=Q,NOTIFY=R3893
14. IF(N1 GT 600) *+1 // NOTIFY=R3893
15. @ N2 = 0
16. READ STRING S4 PROMPT'PDS OR SEQ? '
17. IF('%S4' EQ 'PDS') READ STRING S3 PROMPT'DATA INPUT MEMBER NAME? '
18. IF('%S4' EQ 'SEQ') READ STRING S3 PROMPT'INPUT DATA SET NAME? '
19. READ STRING S1 PROMPT'GSC24.F8XX.DATA FILE NUMBER? '
20. @ S1 = LPAD(S1,2,'0')
21. IF(N2 NE 0) X 31.000
22. READ STRING S5 PROMPT'LIST INPUT DATA SETS? '
23. IF('%S5' NE 'Y') X 31.000
24. *+1 //PRINTA%N2 EXEC PGM=UPRINT,REGION=60K
25. *+1 //SYSPRINT DD SYSOUT=A
26. IF('%S4' EQ 'PDS') *+1 //SYSUT1 DD DSN=R3893.DATA(%S3),DISP=SHR
27. IF('%S4' EQ 'SEQ') *+1 //SYSUT1 DD DSN=R3893.%S3,DISP=SHR
28. *+1 //PRINTB%N2 EXEC PGM=UPRINT,REGION=60K
29. *+1 //SYSPRINT DD SYSOUT=A
30. *+1 //SYSUT1 DD DSN=R3893.GSC24.F8%S1.DATA,DISP=SHR
```

```
31.  @ N2 = N2 + 1
32.  READ VALUE NO PROMPT'# OF PLOTS? '
33.  READ VALUE N6 PROMPT'# OF CYL FOR TEMP DSN? '
34.  @ N4 = N1/60
35.  @ N8 = N1 - 60 * N4
36.  READ STRING SO PROMPT'BIT RATE (INTEGER)? '
37.  IF(SIZ(SO) GT 5) X 36.000
38.  *+1 //GO%SO%N2 EXEC PGM=GASPSIM,REGION=%N5K,TIME=(%N4,%N8)
39.  *+1 //STEPLIB DD DSN=R3893.GASP.LOAD,DISP=SHR
40.  *+1 //FT03F001 DD *
41.  READ VALUE N3 PROMPT'PLOT EVERY XTH POINT? '
42.  READ STRING SO PROMPT'POS OR NEG STUFF? '
43.  IF('%SO' EQ 'POS') @ N1 = -1
44.  IF('%SO' EQ 'NEG') @ N1 = 1
45.  IF('%SO' EQ 'POS') X 48.000
46.  IF('%SO' EQ 'NEG') X 48.000
47.  X 42.000
48.  *+1 %N3,%N1
49.  *+1 /*
50.  *+1 //FT04F001 DD DSN=R3893.GSC24.F8%S1.DATA,DISP=SHR
51.  IF('%S4' EQ 'PDS') *+1 //FT05F001 DD DSN=R3893.DATA(%S3),DISP=SHR,
52.  IF('%S4' EQ 'PDS') *+1 // LABEL=(,,,IN)
53.  IF('%S4' EQ 'SEQ') *+1 //FT05F001 DD DSN=R3893.%S3,DISP=SHR
54.  *+1 //FT06F001 DD SYSOUT=A
55.  *+1 //FT11F001 DD DSN=&GPLOT1,DISP=(NEW,DELETE),SPACE=(CYL,%N6),
56.  *+1 // DCB=(RECFM=VS),UNIT=SYSDA
57.  IF(NO EQ 2) *+1 //FT12F001 DD DSN=&GPLOT2,DISP=(NEW,DELETE),SPACE=(C
58.  IF(NO EQ 2) *+1 // DCB=(RECFM=VS),UNIT=SYSDA
59.  READ STRING SO PROMPT'MORE RUNS? '
60.  IF('%SO' EQ 'Y') X 16.000
61.  *+1 //
62.  LIST %NO/L UNN
63.  READ STRING SO PROMPT'RUN JOB(Y OR N)? '
64.  IF('%SO' EQ 'N') X 71.000
65.  IF('%SO' EQ 'Y') X 67.000
66.  X 63.000
67.  FREE DA(LOAD,GSC24.F8%S1.DATA,DATA)
68.  SUB %NO/L UNN
69.  JO
70.  X 4.000
71.  SET ESC
```

APPENDIX G

STUFF/FILL PROGRAM INPUT DATA

```
5,1
    767    800    833    866    433
    145    107     69     32    526
   1000   1000   1000   1000   1000
1000010000100001000010000
15,894
       1      1      1      1      1      2      2      2      2      3      3      3      4      4
       5
```

APPENDIX H

STUFF/FILL PROGRAM OUTPUT DATA

```
     767
 5    5   10    5    5   10    5    5    5    5    5    5   10    5
 5    5    5    5    5   10    5    5    5    5    5    5   10    5
 5   10    5    5   10    5    5    5    5    5    5   10    5    5
 5    5    5    5   10    5    5    5    5    5    5   10    5    5
10    5    5   10    5    5    5    5    5    5   10    5    5    5
 5    5    5   10    5    5    5    5    5    5   10    5    5    5
 5    5    5   10    5    5    5    5    5    5   10    5    5    5
 5    5    5   10    5    5    5    5    5    5   10    5    5   10
 5    5   10    5    5    5    5    5    5   10    5    5    5    5
 5    5   10    5    5    5    5    5    5   10    5    5    5    5
 5    5   10    5    5    5    5    5    5   10    5    5    5    5
 5    5   10    5    5    5    5    5    5   10    5    5   10    5
 5    9    5    5    5    5    5    5   10    5    5    5    5    5
 5   10    5    5    5    5    5    5   10    5    5    5    5    5
 5   10    5    5    5    5    5    5   10    5    5    5    5    5
 5   10    5    5    5    5    5    5   10    5    5   10    5    5
10    5    5    5    5    5    5   10    5    5    5    5    5    5
10    5    5    5    5    5    5   10    5    5    5    5    5    5
10    5    5    5    5    5    5   10    5    5    5    5    5    5
10    5    5    5    5    5    5   10    5    5   10    5    5   10
 5    5    5    5    5    5   10    5    5    5    5    5    5   10
 5    5    5    5    5    5   10    5    5    5    5    5    5   10
 5    5    5    5    5    5   10    5    5    5    5    5    5   10
 5    5    5    5    5    5   10    5    5   10    5    5   10    5
 5    5    5    5    5    9    5    5    5    5    5    5   10    5
 5    5    5    5    5   10    5    5    5    5    5    5   10    5
 5    5    5    5    5   10    5    5    5    5    5    5   10    5
 5    5    5    5    5   10    5    5   10    5    5   10    5    5
 5    5    5    5   10    5    5    5    5    5    5   10    5    5
 5    5    5    5   10    5    5    5    5    5    5   10    5    5
 5    5    5    5   10    5    5    5    5    5    5   10    5    5
 5    5    5    5   10    5    5   10    5    5   10    5    5    5
 5    5    5   10    5    5    5    5    5    5   10    5    5    5
 5    5    5   10    5    5   10    5    5    5    5    5    5    5
 5    5   10    5    5    5    5    5   10    5    5    5    5    5
 5    5   10    5    5   10    5    5   10    5    5    5    5    5
 5   10    5    5    5    5    5    5    9    5    5    5    5    5
 5   10    5    5    5    5    5    5   10    5    5    5    5    5
 5   10    5    5    5    5    5    5   10    5    5    5    5    5
 5   10    5    5   10    5    5   10    5    5    5    5    5    5
10    5    5    5    5    5    5   10    5    5    5    5    5    5
10    5    5    5    5    5    5   10    5    5    5    5    5    5
10    5    5    5    5    5    5   10    5    5    5    5    5    5
10    5    5   10    5    5   10    5    5    5    5    5    5   10
 5    5    5    5    5    5   10    5    5    5    5    5    5   10
 5    5    5    5    5    5   10    5    5    5    5    5    5   10
 5    5    5    5    5    5   10    5    5    5    5    5    5   10
```

H-1

MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

```
 5      5     10      5      5     10      5      5      5      5      5      5     10      5
 5      5      5      5      5     10      5      5      5      5      5      5      9      5
 5      5      5      5      5     10      5      5      5      5      5      5     10      5
 5      5      5      5      5     10      5      5      5      5      5      5     10      5
 5     10      5      5     10      5      5      5      5      5      5     10      5      5
 5      5      5      5     10      5      5      5      5      5      5     10      5      5
 5      5      5      5     10      5      5      5      5      5      5     10      5      5
 5      5      5      5     10      5      5      5      5      5     30
```

```
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000 10000
```

```
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000100001000010000
10000100001000010000100001000010000100001000010000100001000010000
```

APPENDIX I

GASP PROGRAM INPUT DATA

The input data consists of two data sets, one is the output data from the staff/fill bit program as shown in Appendix H, the other is listed below.

```
J J CORMACK          24    12   27 1979    100000000000011
        2     8    0    0    1    0    25    4    1  200   30    0    10
                    1PHMAX
                    2PHMIN
                    1PHOUT
                    2PNODE
                    3PSLEW
                    4PHERRN
                    5PHTRK
                    6PHSRCE
                    7PTRK
                    8PSLEWT
                    1TIME              11    6    0         0.0
                    1DDEC          1    1         0.0         1.0
                    2SSYNC         1    1         0.0       520.0
                    3OUTPUT        1    1         0.0       360.0
                    4ISOURCE       2    2      -360.0       360.0
                    5EPHERRN       1    1      -360.0       360.0
                    6TPHTRK        1    1        36.0        36.0
        2
        2
        4    0     1.0E-7      1.0E-8       1.0E-8      1.0E-4       -1.0
        0    1     1              0.0                    1
        0
        1         7813            5
9.58907E-3         1
        4          2
        9.6E-3          09.58907E-3
        0.0        90.0
1.8293E-1    .230107
9.6001E-3          09.58907E-3
        0.0        90.0
1.8293E-1    .230107
9.5999E-3          09.58907E-3
        0.0        90.0
1.8293E-1    .230107
        9.6E-3          09.58907E-3
        0.0        90.0
1.8293E-1    .230107
        9.5E-3
        2887.0          2.0
```

APPENDIX J

GASP PROGRAM OUTPUT

This output consists of a time domain waveform plot of several parameters in the multiplexer's smoothing loop.

SIMULATION PROJECT NUMBER 24 BY J J CORMACK

GASP IV VERSION 25JAN75

DATA COLLECTION STARTS AT FRAME 1 AND STOPS AT FRAME 40 WITH 5 SAMPLES PER FRAME

SYSTEM PARAMETERS FOR 1 NODES
ALL NODES ARE TYPE 2 SYSTEMS

NODE ADDR LISTED
1000 1000 1000 1000 1000 1000 1000 1000 1000 1000

THIS DELAY LISTED
1000100001000100001000010000010000010000100001000010000

NLPR ARRAY LISTED
1 1 1 1 1 1 1 1 1 1 1

NODE 1

V1 FREQUENCY SOURCE = 9.5930686E-03 MHZ
NUMBER OF BITS PER OPPORTUNITY = 651
PHASE - 4TH DATA POINT IS PLOTTED

V2 FREQUENCY SOURCE = 9.5939945E-03 MHZ
CORRECTION APPLIED AT OUT OF LF FRAME
VCF NOMINAL FREQUENCY = 6.5890886E-03 MHZ

V2 INITIAL PHASE = 0. DEG
NODE OUTPUT PHASE = 270. DEG

NATURAL FREQUENCY (PN) = 1.8282999E-01 HZ
DAMPING (ZETA) = 3.614

TRACKING LOOP PARAMETERS
VCF NOMINAL FREQUENCY = 9.5890886E-03 MHZ

NATURAL FREQUENCY (PN) = 2.0070000E+03 HZ
DAMPING (ZETA) = 2.000

J-3

GASP SUMMARY REPORT

STUDENT ON PROJECT NUMBER 28 BY J J CONRACK

DATE 11/ 17/ 1979   RUN NUMBER   1 OF   1

CURRENT TIME = 0.3187E+01

STATISTICS FOR VARIABLES BASED ON OBSERVATIONS

	MEAN	STD DEV	SD OF MEAN	CV	MINIMUM	MAXIMUM	OBS
PUMAX	0.3001E+03	0.2112E+02	0.3696E+01	0.6251E-01	0.2980E+03	0.3000E+03	36
PUMIN	0.3013E+03	0.3463E+00	0.6468E+01	0.1146E-00	0.2145E+03	0.3000E+03	30

STATISTICS FOR TIME-PERSISTENT VARIABLES

	MEAN	STD-DEV	MINIMUM	MAXIMUM	TIME INTERVAL	CURR. VALUE
PUCW	-0.3079E+03	0.2485E+02	0.0	0.3608E+03	0.3187E+01	0.2236E+03
PSLFM	-0.8057E+02	0.1305E+01	-0.2259E+01	0.6406E+01	0.3187E+01	0.6076E+00
PUUTM	-0.1862E+03	0.1158E+03	0.0	0.1340E+03	0.3187E+01	0.8106E+02
PUTFK	0.9759E-04	0.8317E-04	0.0	0.3633E-03	0.3187E+01	0.6968E-06
PUFRCB	0.0	0.0	0.0	0.2641E-03	0.3187E+01	0.2641E-03
PUT-K	0.4802E+00	0.4427E+00	-0.4092E+01	0.5955E+01	0.3187E+01	0.5121E+00
PSLCUT	-0.6071E+05	0.9435E+15	-0.4019E+06	0.1273E+04	0.3187E+01	0.9000E+02

J-4

STORAGE FILE STORAGE AREA DUMP AT TIME 0.3187E+01

MAXIMUM NUMBER OF ENTRIES IN FILE STORAGE AREA = 6

PRINTOUT OF FILE NUMBER 1

F*KO = 0.3187E+01
001142 0.3187E+01

THE DEPTED PER STATISTICS 0.3187E+01
AVERAGE NUMBER IN FILE   3.0007
STANDARD DEVIATION   1.4143
MAXIMUM NUMBER IN FILE   6

		FILE CONTENTS		
ENTRY	1 =	0.3187E+01	0.1007E+01	0.0
ENTRY	2 =	0.3201E+01	0.1000E+01	0.0
ENTRY	4 =	0.321?E+01	0.1000E+01	0.0
ENTRY	5 =	0.3270E+01	0.1000E+01	0.0
ENTRY	6 =	0.325?E+01	0.2000E+01	0.0

J-5

*GASP STATE STORAGE AREA DUMP AT TIME 0.3187E+010

(I)	SS(I)	DD(I)
1	0.3290E+00	0.6000E+22
2	0.4751E+01	0.1644E+02
3	0.4751E-01	0.1322E+01
4	0.2424E-04	0.1466E-08
5	0.3143E-07	0.1907E-05

TABLE NUMBER 100
RUN NUMBER 1

TIME	DEC	SYNC	OUTPUT	SOURCE	PHERRN	PHTRK
MINIMUM	0.0	0.0	0.2124C+C3	0.0	-0.2803E+03	0.0
MAXIMUM	0.1000E+01	0.5515E+03	0.3900E+03	0.0	0.2223E+03	0.2461F-03

J-7

J-10

OUTPUT CONSISTS OF 235 POINT SETS (1410 POINTS)

J-11

DISTRIBUTION LIST

STANDARD:

R100 - 2	R200 - 1
R102/R103/R103R - 1	R300 - 1
R102M - 1	R400 - 1
R102T - 9	R500 - 1
R104 - 1	R700 - 1
R110 - 1	R800 - 1
R123 - 1 (Library)	NCS-TS - 1
R124A - (for Archives)	101A - 1
	312 - 1

R102T - 12

DCA-EUR - 2 (Defense Communications Agency European Area
            ATTN: Technical Library
            APO New York 09131)

DCA-PAC - 3 (Commander
            Defense Communications Agency Pacific Area
            Wheeler AFB, HI 96854

DCA SW PAC - 1 (Commander, DCA - Southwest Pacific Region
            APO San Francisco, CA 96274

DCA NW PAC - 1 (Commander, DCA - Northwest Pacific Region
            APO San Francisco, CA 96328

DCA KOREA - 1 (Chief, DCA - Korea Field Office
            APO San Francisco, CA 96301

DCA-Okinawa - 1 (Chief, DCA - Okinawa Field Office
            FPO Seattle 98773

DCA-Guam - 1 (Chief, DCA - Guam Field Office
            Box 141   NAVCAMS WESTPAC
            FPO San Francisco 96630

US NAV Shore EE PAC - 1 (U.S. Naval Shore Electronics Engineering
                        Activity Pacific, Box 130, ATTN: Code 420
                        Pearl Harbor, HI 96860

1843 EE SQ - 1 (1843 EE Squadron, ATTN: EIEXM
            Hickam AFB, HI 96853

DCA FO ITALY - 1 (DCA Field Office Italy, Box 166
            AFSOUTH (NATO), FPO New York 09524

(continued)

DISTRIBUTION LIST (cont'd)

USDCFO - 1 (Chief, USDCFO/US NATO
             APO New York 90667)

SPECIAL:

DCA   Code 282
           231

Commander, Rome Air Development Center,
ATTN: DCLD, Griffiss AFB, NY 13442

Commander, AFCC, ATTN: 1842 EEG/EETTC,
Scott AFB, IL 62225

Commander, USACEEIA, ATTN: CCC-CED-XEM,
Ft. Huachuca, AZ 85613

Commander, ESD, ATTN: DCF-1, Hanscom Field, MA 01730

Commander, USASATCOMA, ATTN: DRCPM-SC-9E,
Ft. Monmouth, NJ 07703

FILM

6